Interaction on emotions

Arno Hartholt Tijmen Joppe Muller

January 16, 2004

Authors
Arno Hartholt, d.o.a.hartholt@student.utwente.nl
Tijmen Joppe Muller, tijmen@avpec1910.nl

Supervisor
Jonathan Gratch, gratch@ict.usc.edu
Institute for Creative Technologies
13274, Fiji Way
Marina del Rey, California 90292-7008
United States of America

Mentor
Anton Nijholt, anijholt@cs.utwente.nl
University of Twente
Postbus 217
7500 AE Enschede
The Netherlands

Acknowledgement
This work was funded by the Department of the Army under contract DAAD 19-99-D-0046. Any opinions, findings, and conclusions expressed in this article are those of the authors and do not necessarily reflect the views of the Department of the Army.

Preface

A compulsary part within the study of Computer Science at the University of Twente is a 14 week internship. We've chose to perform this internship abroad, at the Institute for Creative Technologies in the United States of America. The internship consists of two assignments, both within the Mission Rehearsel Exercise project. The first assignment, *Interaction on emotion*, will be discussed in this report; the other assignment, *Everything in perspective*, is the subject of a second report.

We had a great time during time during our internship, both on a personal[1] and professional level. In terms of this particular assignment we want to thank Jonathan Gratch for making this internship possible and for taking the time and effort to guide us. Also, we want to thank David Traum for all the times he explained how natural language architecture was designed and implemented, and especially for helping debug our code. Thanks to Stacy Marsella for the discussions about emotion dialogue and coping strategies. Finally, we want to thank Anton Nijholt for his valuable insights and his guidance during our internship.

Arno Hartholt and Tijmen Muller
January 16, 2004

[1] Working and lunching, for example, was made more enjoyable because of Charlie, who wouldn't leave our side. We tried to reason with him. We tried to convince him he was beginning to annoy us. He still wouldn't leave our side.

Abstract

This report describes the addition of an emotion dialogue to the Mission Rehearsal Exercise (MRE) system. The goal of the MRE system is to provide an immersive learning environment for army officer recruits. The user can engage in conversation with several intelligent agents in order to accomplish the goals within a certain scenario. Although these agents did already posses emotions, they were unable to express them verbally. A question - answer dialogue has been implemented to this purpose. The implementation makes use of proposition states for modelling knowledge, keyword scanning for natural language understanding and templates for natural language generation. The system is implemented using Soar and TCL.

An agent can understand emotion related questions in four different domains, *type*, *intensity*, *state*, and the combination of *responsible-agent* and *blameworthiness*. Some limitations arise due to the techniques used and to the relative short time frame in which the assignment was to be executed. Main issues are that the existing natural language understanding and generation modules could not be fully used, that very little context about the conversation is available and that the emotion states simplify the emotional state of an agent. These limitations and other thoughts give rise to the following recommendations for further work:

- Make full use of references.
- Use coping strategies for generating agent's utterances.
- Use focus mechanisms for generating agent's utterances.
- Extend known utterances.
- Use NLU and NLG module.
- Use emotion dialogue and states to influence emotions.
- Fix known bugs.

Keywords: *natural language, dialogue, emotions, multi-agent systems, keyword scanning, templating*

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This document is the report of one of two internship assignments – the assignment is explained in chapter 3.1. This chapter gives a brief introduction to the Mission Rehearsel Exercise and the current implemented scenario. A more extensive explanation of the system will be given in chapter 2. Chapter 3 gives the requirements of our assignment. In chapter 4, we analyse the parts of the system that are important to our assignment and mention the datastructures we need to use or design. The design and implementation are given in respectively chapter 5 and chapter 6. The testing phase will be discussed in chapter 7. The conclusion and recommendations can be found in chapters 8 and 9. The appendices contain the possible speech acts, written code and test plans.

Figure 1.1: Screenshot of the Bosnia scenario

1.1 Mission Rehearsel Exercise project

The goal of the *Mission Rehearsal Exercise* system is to provide an immersive learning environment where the participants experience the sights, sounds and circumstances they will encounter in real-world scenarios while performing

mission-oriented training [ICT03]. The user can engage in conversation with several intelligent agents in order to accomplish the goals within a certain scenario. We will explain more about the system in chapter 2.

1.2 Bosnia scenario

In the *Bosnia scenario* the user is a young lieutenant in the US Army on his first peacekeeping mission. As you are sent to help another group of soldiers, called Eagle 1-6, inspect a suspected weapon cache in a town called Celic, you find that an accident occured at the assembly area: a civilian car crashed into one of your platoon vehicles. A local boy lies injured on the ground, with his mother and a medic from your team next to him. In the background a group of locals is gathering and unrest is rising. A sketch of the situation is drawn in figure 1.2.

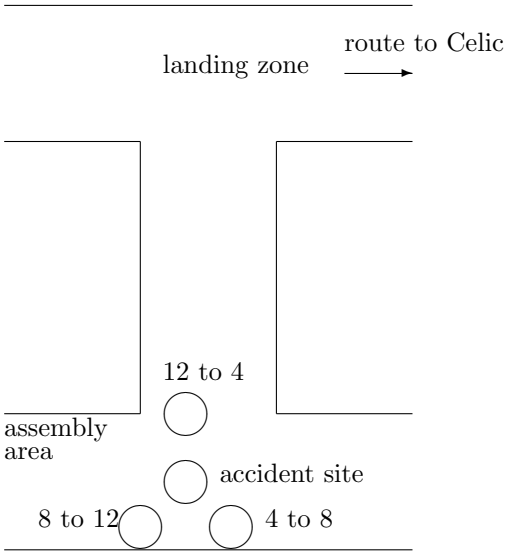

Figure 1.2: The situation in the Bosnia scenario

It is up to you to decide what your team, Eagle 2-6, is going to do. You have the possibility to move along to Celic or try and help the boy. One way is to call an ambulance, another is to call the emphMedEvac, a helicopter, from your base. Either way, you need to interact with your teammembers to find a solution.

As the user you are able to interact with three emphintelligent agents: the sergeant, the medic and the mother of the injured boy. [Ric02]

Chapter 2

Background

The Mission Rehearsal Exercise (MRE) project tries to create an immersive and interactive learning environment in which users can gain experience through decision-making scenarios [Ric02], like the Bosnia scenario described in chapter 1. An important part of making such an environment both believable and useful is the use of intelligent agents that cohabit virtual worlds with people, and support face-to-face dialogue, serving as guides, mentors and teammates. In the Bosnia scenario, the sergeant, medic and mom are implemented as such agents.

The whole system tries to immerse the user by displaying the visuals on an eight-foot-tall screen in a 150-degree arc with a 12-foot radius. The graphics are rendered with Multigen-Paradigm's Vega. Immersive audio software uses 10 audio channels and two subwoofer channels to envelop a participant in spatialized sounds that include general ambience (such as crowd noise) and triggered effects (such as explosions or helicopter flyovers).

This chapter provides the reader with a theoretical and practical background concerning the MRE technology. In section 2.1 we talk briefly about it's architecture. Next, we discuss *Steve*, the result of former research by the University of Southern California's Information Sciences Institute, which forms the basis for the MRE agents. As emotions and natural language are of particular interest for this assignment, these are discussed separately in sections 2.3 and section 2.4 respectively.

2.1 MRE architecture

MRE seeks to integrate and expand several results form various fields of research, like embodied agents, natural language and emotion. Figure 2.1 gives an overview of it's architecture.

The MRE system consists of a virtual world and it's agent inhabitants. The world is modeled by a set of *sequence files* wherein states and state changes are defined. These files can be called both by the system and the system's operator using a GUI. The agents use speech recognition software (Sonic) and an in-house developed Natural Language Understanding module. The core of the agents is formed by *Soar* code, which models perception, planning, emotion, dialogue, natural language generation and action. Soar is a low-level reason-

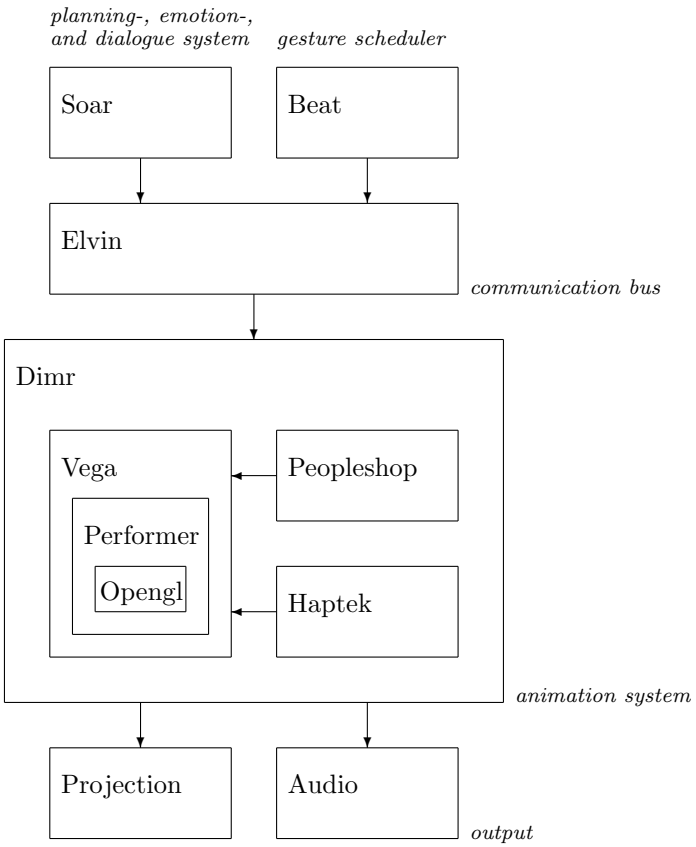

Figure 2.1: Architecture of the MRE system

ing programming language, which builds an environment by creating very small datablocks by the name of *working memory elements* (WME). These WME's contain very small pieces of information about the world. By the use of operators, which are defined by the programmer himself, it is possible to reason about the world [Lai99]. The Soar code makes up Steve, wich will be discussed in more detail in section 2.2.

Steve's text output is synthesized into speech by Festival. This module works together with the Beat module which creates gestures to accompany the speech. The sound and visuals of both the world and the agents are generated through some audio protocols and DIMR. DIMR is the visualization module, consisting of Vega (renders the environment and special effects), PSERT (People Shop TM Embedded Runtime System, responsible for animating the agents) and Haptek (facial animations). All the modules communicate via Elvin/XML messages through a shared communication bus.

2.2 Steve

Steve is the result of the University of Southern California's Information Sciences Institute research and is capable of collaborating with people in 3D virtual worlds as an instructor or teammate. It is able to provide feedback to students and can lead students around in the virtual world.

Steve's cognitive basis is formed by a domain-independent layer, constructed in Soar. Soar is a general model of human cognition and as such provides Steve with basic cognitive capabilities. These basics are enhanced to support task-oriented collaboration, like demonstration and conversation. On top of this layer, a declarative representation of domain tasks can be placed. For a more extensive introduction to Steve, see [Ric00].

For the MRE project, Steve had to be extended in a variety of ways. We'll shortly discuss *perception*, *virtual bodies* and *task planning* here, emotions and natural language are described in the following sections.

2.2.1 Perception

In order to make an agent believable as a virtual human, it has to *perceive* his environment the same way a real human being should. Originally, Steve was omniscient; he received messages from the virtual world simulator describing every state-change relevant to his task model, regardless of his current location or attention state. In MRE, the Steve agent's perception is more human-like, following the research presented in [Hil99], [Hil00] and [Cho99]. His vision is limited to 190 horizontal degrees and 90 vertical degrees. The level of detail Steve perceives about objects is high, medium, or low, depending on where the object is in his field of view and whether he is giving attention to it. Steve can perceive both dynamic and static objects in the environment. The first are perceived under the control of a simulator, by filtering updates that the system periodically broadcasts. Static objects such as buildings or trees are perceived by using the scene graph and edge-detection and are encoded in a cognitive map.

2.2.2 Virtual body

Steve's is a so-called *emobied conversational agent*, which is a type of *virtual body* that, next to speech, uses *nonverbal communication* like facial expressions, gestures and body stance when engaging in conversation. For an introduction to embodied conversational agents, see [Cas00].

Steve was designed to accommodate different bodies. His motor-control module accepts abstract motor commands from his cognition module and sends detailed commands to his body through a generic API. Steve's original body, although suitable for tutoring ends, had some limitations which restricted it from simulating a life-like human being. It had, for instance, no legs, and it's gestures and facial expressions where somewhat limited. For MRE, new bodies were developed by Boston Dynamics Incorporated with faces developed by Haptek Incorporated. In order to provide the agents with a realistic body without losing flexibility over their motions, a method using both motion capture and procedural animation was used. Motion capture provided a basic repertoire of gestures, which were then decomposed into stages (such as preparation, stroke,

and retraction). Moreover, these gestures are available in two extremes: a small, restrained one and a large, emphatic one. A Steve agent can dynamically generate any gesture between these extremes by specifying a weighted combination of the two. Thus, bodies leverage the realism of motion capture while providing the flexibility of procedural animation.

2.2.3 Task planning

In a system with agents serving as teammates, an agent should be able to do some goal focused planning: *task modelling*. For a believable world, the user should be able to act freely, within the restrictions of the scenario, e.g. taking the goals of the scenario into account. This means the user has to be free in decision making, trying different alternatives, and interact naturally with teammates. As a result, the agents need to be able to understand authority, responsibility, coordinated actions, organizational relationships and most of all the structure of tasks.

The plan representation, which is a relatively standard hierarchical model, consists of a set of steps, either primitive (i.e. a physical or sensing action) or abstract (i.e. a task that needs further decomposing), a set of ordering constraints, and a set of causal links and threat relationships that define the dependencies among the steps. In this way, a causal link can define one step as either a precondition or a threat to another step. In the MRE project, the agents use domain-independent reasoning algorithms over general representation of team tasks to create a task model. In order to create such a model, the agent uses the implemented tasks and world knowledge.

In addition to the understanding of the tasks, the agents also need to understand the social relationships or *roles* among themselves. One thing is that the task steps are linked to an agent responsible for that task. Another is the optional *authorizing agent* – a responsible agent cannot execute his task before the authorizing agent has granted his authority; this is, of course, very common in the military.

Given an abstract task for the whole team to accomplish, each agent independently uses his task knowledge to create his task model. This practically means decomposing the abstract task recursively until a task model has been created. Since agents may not have the same knowledge, different task models may occur for different agents. As a result, *negotiating* may be needed – here, natural language comes into focus, with the different roles in the background [Tra03] [Ric02].

2.3 Emotions

One of the main goals of the MRE project is implementing a computational model of human behavior. Emotions are considered a very important part of artificial intelligence, since they have a great influence on human behavior, are needed to correctly interpret beliefs, motives and intentions, and play an important role in social communication. Of course, emotions are tightly connected to other capabilites of these *virtual humans*, such as planning, acting, natural language understanding, and speech. Creating a domain-independent model could assist in building believable intelligent agents in general. [Gra04].

2.3.1 Theoretical framework

Attempting to find an unified theory, the MRE project characterizes emotion as the result of *cognitive appraisal*. Cognitive appraisal emphasizes the connections between cognition, emotion, personality and the resulting coping strategies, following certain psychological theories.

Strategy	**Description**
active coping	taking steps to remove or circumvent the stressor
planning	coming up with action strategies
seeking social support	seeking advice, assistance, or information

Table 2.1: Problem-focused coping strategies

The sources for all emotional behavior are the environment and the goals and beliefs of an agent. Appraisal can be seen as the interpretation of the *person-environment relationship*, assigning values to abstract attributes of this relationship, called *appraisal variables* – these are listed in table 4.1. The environment partly consists of other agents; this is represented by the fact that an agent may have beliefs about the feelings of other agents. The appraisal variables' values depend on the cognitive processes that build up the individual's interpretation of how (external) events influence their goals and beliefs.

The resulting *coping strategies* plan actions to change either the environment (*problem-focused strategies*) or the agent's goals and beliefs (*emotion-focused strategies*). Some common coping strategies are listed in table 2.1 and table 2.2. What strategy is picked by an agent is based on his 'personality': for example, a self-confident agent would use a planning strategy to solve a problem, while a not so confident agent could use a denial strategy. The system is explained graphically in figure 2.2.

Strategy	**Description**
suppress competing activities	put other projects aside or let them slide
restraint coping	waiting for appropriate opportunity, holding back
seeking social support	getting moral support, sympathy, or understanding
positive reintegration	look for silver lining and try to grow as a person
acceptance	accept stressor and learn to live with it
denial	denying the reality of event
behavioral disengagement	admit you cannot deal and reduce effort
mental disengagement	daydreaming, sleeping, turn to drugs or religion

Table 2.2: Emotion-focused coping strategies

The following is a simplified algorithm for cognitive appraisal by the name of the EMA *algorithm*. It is explained in detail in [Gra04].

1. Construct and maintain a causal interpretation of ongoing world events in terms of beliefs, disires and intentions.

2. Generate multiple appraisal frames that characterize features of the causal interpretation in terms of appraisal variables.

3. Map individual appraisal frames into individual instances of emotion.

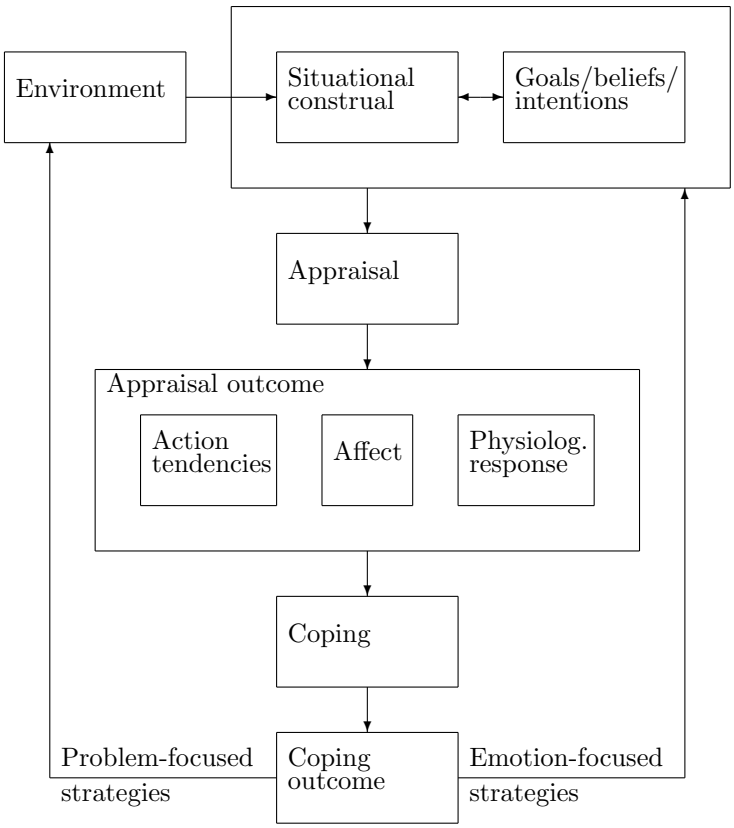

Figure 2.2: The cognitive-motivational-emotive system

4. Aggregate emotion instances into a current emotional state, focusing on instances associated with recent changes to the causal interpretation.
5. Adopt a coping strategy in response to the current emotional state.

2.3.2 So, what is actually implemented?

To implement a realistic emotional behavior in an interactive setting, the coping strategies need to be dynamically dependent of the underlying emotional system. This means the internal processes of an agent in the MRE system need to lead to facial expressions, gestures, intonation and action planning. A realisation needs to satisfy various requirements:

- To make dynamic interaction possible on cognition, appraisal and coping, the model must represent intermediate knowledge states that can be appraised.
- To reason about relevance and desirability, the model must represent preferences over outcomes.
- To make causal attributions, causal relations between states need to be possible.

- To reason about likelihood and expectedness about future states, the system needs to represent factors influencing events, outcomes and interaction between these events.
- To reason about urgency, the system needs to have some notice of time.
- To reason about controllability, the system must represent to what extent events can be controlled.
- To reason about social power, the system need to represent relations and organizational hierarchies of agents.
- To reason about adaptability and to support emotional coping strategies, the system must represent subjective beliefs.
- To reason about ego-involvement, the system needs to represent how 'egoistic' a desire of an agent is.

In order to meet these requirements, the emotional system of MRE is based on the design of Steve (section 2.2, which in turn is built using Soar. This makes it possible to, for example, reason about a certain event that is about to happen and that endangers the personal desires (or desired *states*) and as a result leads to appraised fear. More details about the way the emotional state of an agent is constructed in the MRE system is presented in section 4.1.

The emotional state and the coping strategies together result in mental and physical activity. At this point, these states are linked to the movement of the virtual body, so the agent adjusts his facial expressions and gestures to what he feels. Manipulation of the utterances (different utterances for one speech act and adjusted intonation, depending on emotion) has only been partially implemented, but the foundation to realize the whole of it is present.

2.4 Natural language

Steve used commercial speech recognition and synthesis products to communicate with human students and teammates. It had no true natural language understanding capabilities and understood only a relatively small set of preselected phrases. Although easy to implement, this method has two major drawbacks. The obvious one is that a system with hard coded language is very limited in both understanding and generating speech, and that adding new content is a cumbersome task. Secondly, all the recognition is done by the speech recognizer itself rather than by modules that have access to the evolving task and dialogue context. Because of these reasons, Steve has been modified in a variety of ways. We distinguish three areas, namely *dialogue*, *natural language understanding* (NLU) and *natural language generation* (NLG). These will be discussed in the following subsections.

2.4.1 Dialogue

In MRE, the main conversation is between the lieutenant and the sergeant. Also, the medic is some times brought in, and the mother is an important overhearer. Additionally, separate conversations between the sergeant and the

squad leaders occur, and both the lieutenant and the sergeant can engage in some radio conversation. The agents must be capable to reason about who they are talking to, who is listening, and whether they are being addressed.

The *dialogue model* used within MRE is capable of accomplishing these multi-party, multi-conversation dialogues. It is modelled as a *layered structure*:

- contact
- attention
- conversation
  - participants
  - turn
  - initiative
  - grounding
  - topic
  - rhetorical
- social commitments (obligations)
- negotiation

The state of each layer is represented by an *information state* and a set of *dialogue acts* corresponding to changes to this information state. There are also conventional *signals* – behaviors (for instance a speech act or gazing) that can be associated with the performance of dialogue acts, given the right context.

The *contact layer* is concerned with which individuals are accessible for communication. The *attention layer* concerns the object or process that agents attend to. Contact is required for attention. The *conversation layer* models the separate dialogue episodes that go on during an interaction. This layer has several sub-layers which may consist different information for each conversation that is going on. The *participants* include active speakers, adressees, or overhearers. The *turn* indicates which participant has the right to speak. *Initiative* holds the person who is leading the conversation. The *grounding* component tracks how information is added to the common ground of the participants. The *topic* contains the topic of the associated conversation and furthermore there are *rhetorical* connections relating content units to each other. Once material is grounded, even as it still relates to the topic and rhetorical structure of an ongoing conversation, it is also added to the social fabric linking agents, which is not part of any individual converstaion. This includes *social commitments* – both obligations to act or restrictions on actions, as well as commitments to factual information. There is also a *negotiation* layer, modeling how agents come to agree on these commitments. A more in-depth survey of these layers can be found in [Tra02].

In addition to these layers, there are several classes of *rules* which relate the dialogue acts to the information state. These are:

Recognition rules that associate observed behavior (speech and/or other modalities) with performance of one or more of these dialogue acts.

Update rules that modify the information state components with information from the inferred dialogue acts.

Selection rules that decide which dialogue acts the system should perform.

Realization rules that indicate how to realize the selected dialogue act, using natural language, non-verbal communications, and other behavior.

Currently, the model isn't implemented entirely. Some aspects, like the rhetorical layer are divided amongst several layers and others demand more attention. This is especially true for the contact and attention layer.

2.4.2 Natural language understanding

The agent relies on external speech recognition and the NLU module. The goal of this module is to convert a string of words, as uttered into the microphone by the MRE trainee and then recognized by the speech recognition module, into one or more *semantic frames* that represent the meaning of the sentence. This module currently includes three engines operating in parallel: two use finite state technology and one is a statistically trained parser. These three engines exhibit complementary strengths. The first finite state engine is limited in scope, but its output is always 100 percent correct; it is most useful for sentences where speech recognition is fully correct. The second can recognize portions of inputs, for cases in which speech recognition provides a partially correct sentence. The third, statistical engine is more robust, and can easily be extended to handle new sentence types and new words, but its output may include erroneous propositions. This module is stand-alone Java code.

The semantics form the input for the Soar code which keeps track of the dialogue and is responsible for the natural language generation. Knowledge is represented using *state propositions*, which form the basis for the system's reasoning capabilities. The states currently known to the agents can be found in tables 4.2 and 4.3. Two abbreviated examples can be seen below. The first is a state proposition, stating that the boy is currently not healthy. Example two is a partial proposition: the q-slot field represents the missing information, e.g. the attribute that is asked about. Here, the question "Who is hurt?" is represented [Tra03c].

1. `^attribute health-status ^object-id boy ^polarity negative ^time present ^type state ^value healthy`

2. `^q-slot object-id ^attribute health-status ^polarity negative ^time present ^type state ^value healthy`

2.4.3 Natural language generation

Depending on the information state of the dialogue, the Soar code concerning language generation proposes one or more *communicative goals*. Once a goal has been selected the following phases will be executed:

Content selection phase during which is determined whether the communicative goal can be met and what content satisfies the given goal.

Sentence planning phase during which is decided in what way the message is best conveyed. This generates the sentence in an abstract way.

Realization phase during which the abstract sentence is mapped to words and phrase structures.

Ranking phase during which the possibly multiple ways of realizing the sentence is being considered and a best match is selected.

This final sentence is then augmented with communicative gestures, including lip synch, gaze, and hand gestures, converted to XML, and sent to the synthesizer and rendering modules to produce the speech. After the message is spoken, the agent receives feedback in order to verify if the sentence was uttered successfully after which the dialogue state is updated. The article [Tra03b] discusses the NLG phases in more detail.

Chapter 3

Requirement specification

3.1 Assignment

At this point, the agents within MRE do possess emotional behavior, but they are unable to express it explicitly; it surfaces only by means of body language and facial gestures. Our task is to specify and implement utterances that express the emotional state of the agent and the reasons for that state, i.e. the user should be able to ask the agent how it feels and why he is feeling that way.

3.2 Requirements

The final result should be a scenario-independent implementation of an emotional dialogue, where agents can:

- understand user's utterances concerning emotions;
- use existing knowledge of their emotional state to express themselves; and
- express themselves using natural language.

An emotional dialogue should involve the following possible subjects:

- type of emotion;
- intensity of emotion;
- cause of emotion;
- involved agents for a certain emotion.

Chapter 4

Analysis

In this chapter the parts of the system that are important to our assignment are analysed. First, the emotional state is discussed, followed by the agents personality. Next the possible emotion based speech acts are analysed. This chapter concludes with a short description of natural language understanding (NLU) and natural language generation (NLG).

4.1 Emotional state

The *emotional state* of an agent is the result of the *cognitive appraisal* of the world; it is calculated by taking various events into consideration. Some events in the world have a direct impact on the agent's emotional state (i.e. the joy at winning a game) and some events are involved through a causal relation (i.e. feeling guilty for the distress of one's opponent after winning a game). All these appraisals are annotated by attributes, called *appraisal variables*; these variables characterize the event and as a result, determine the type and intensity of the emotion. The different appraisal variables are described in table 4.1; the variables that are actually implemented in the current model are marked with a †. [Gra04]

	Appraisal variable	**Description**
†	relevance	does the event require attention or adaptive reaction
†	desirability	does the event facilitate or threat the agent's goals
†	agency	what agent is responsible for the event
†	blameworthiness	does the responsible agent deserve blame or credit
†	likelihood	how probable is the event to happen
†	inexpectedness	was the event predicted from past knowledge
	urgency	will delaying a response make matters worse
	ego involvement	how does the event impact a person's sense of self
†	controllability	to what extent can the event be influenced
	power	what power has an agent to (in)directly control the event
	adaptability	can the person live with the consequences of the event
†	changability	will the effect change of it's own accord

Table 4.1: Appraisal variables

In the MRE system an appraisal is linked to a certain state and an event (or act) that either helps or threatens progress towards the state. The emotion raised depends on whether the agent wants that state to be true in the world or not – this means the *desirability* variable in table 4.1 is a result of the desirability of the state and whether the event is either helping or threatening that state.

The appraisals (or appraisal frames) in the MRE system are stored in working memory elements (WME's), the elements that contain all the current knowledge in the world in the Soar system. In this section all information available on emotions will be presented; this information can be used to construct speech acts. The information stored in the WME about each emotion are *type* and *intensity* and what event it was raised by. Information on events are the *state*, $importance_{state}$, $desirability_{state}$ (which actually is a binary mapping of the $importance_{state}$ attribute), *likelihood*, *responsible_agent*, *blameworthiness*, $influence$ (i.e. if the event helps or threatens the state, this is the actual type of the appraisal), and *status* (i.e. has it already happened or is it about to happen).

4.1.1 Emotions

There are seven *emotion types*: `joy`, `hope`, `distress`, `fear`, `anger`, `guilt`, and `anxiety`; they are stored in WMEs by the same name. Every emotion has a certain *intensity* between `0` and `1`.

For every appraisal there is an action towards a certain *state* that has happened or is about to happen. All these states in the MRE Bosnia mission are split into states with respect to the mother's perspective, which are listed and explained in table 4.2, and states with respect to the lieutenant's, sergeant's and medic's perspective, which are listed and explained in table 4.3.

State	**Meaning**
`child-dead`	My child is dead
`child-healthy`	My child is healthy.
`troops-helping`	The foreign troops are helping us.
`facilities-ok`	The facilities to help the boy are available.
`lz-secure`	The landing zone for the medivac is secure.
`medevac-called`	The medevac has been called.
`authority-present`	There is authority present at the accident site.
`help-requested`	Help for my son had been requested.

Table 4.2: States in mother's perspective

The attribute $importance_{state}$ has a value between `-100` and `100`. The $desirability_{state}$ is a binary mapping of the former attribute: if $importance_{state} < 0$ then $desirability_{state}$ = `undesired` and if $importance_{state} \geq 0$ then $desirability_{state}$ = `desired`.

The possible values for $influence$ are `facilitator`, which means the emotion is raised by an event that helps progress towards a certain state, and `inhibitor`, which threatens progress.

What type of emotion is raised by an event depends on the *desirability* of the event. This *desirability* depends on the $importance_{state}$ and the $influence$. If

Name	Meaning
maintain-goodwill	The goodwill of the crowd is maintained
crowd-angry	The crowd is angry
boy-dead	The boy is dead
boy-healthy	The boy is healthy.
minor-injuries	The boy has minor injuries.
serious-injuries	The boy has serious injuries.
critical-injuries	The boy has critical injuries.
driver-healthy	The driver is healthy.
driver-minor-inj	The driver has minor injuries.
mother-healthy	The mother is healthy.
know-boy-health	The health of the boy is known.
sqds-in-transit	The squads are in transit to the assembly area.
lt-in-transit	The lieutenant is in transit to the assembly area.
lt-at-aa	The lieutenant is at the assembly area.
lt-at-celic	The lieutenant is at Celic.
sgt-at-aa	The sergeant is at the assembly area.
medic-at-aa	The medic is at the assembly area.
at-boy-aa	The boy is at the assembly area.
mom-at-aa	The mother is at the assembly area.
1st-sqd-at-aa	First squad is at the assembly area.
1st-sqd-at-celic	First squad is at Celic.
1st-sqd-at-lz	First squad is at the landing zone.
	(same for 2nd, 3rd and 4th)
1st-sqd-in-transit	First squad is in transit.
4th-sqd-in-transit	Fourth squad is in transit.
at-boy-hospital	The boy is at the hospital.
mom-in-intersect	The mother is at the intersection.
sqd-in-intersect	The squad is at the intersection.
medevac-at-aa	The medevac is at the assembly area.
medevac-at-base	The medevac is at the base.
medevac-called	The medevac has been called for.
medevac-overhead	The medevac is overhead.
amb-at-aa	The ambulance is at the assembly area.
amb-at-base	The ambulance is at the base.
ambulance-called	The ambulance ahs been called for.
secure-route	The route to Celic has been secured.
aa-secure	The assembly area has been secured.
accident-secure	The accident site has been secured.
12-to-4-secure	12 to 4 hours secure.
4-to-8-secure	4 to 8 hours secure.
8-to-12-secure	8 to 12 hours secure.
lz-secure	The landing zone has been secured.
lz-marked	The landing zone has been marked by green smoke.
lz-clear	The landing zone is cleared of civilians.
support-1-6	Eagle 2-6 is supporting Eagle 1-6
retain-mass	Eagle 2-6 is together (as a team).
fracture-unit	Eagle 2-6 has been split up
1-6-at-celic	Eagle 1-6 is at Celic
hospital-at-tuzla	The hospital is at Tuzla

Table 4.3: States in sergeant's, lieutenant's and medic's perspective

the state is not desired and the event is threatening progress towards that state, the event will of course be desired, i.e. $desirability > 0$. The categorization of the emotions is described in table 4.4; we consider the source of the event known (i.e. $state \neq NULL$) unless stated otherwise.

Joy and hope are the results of a positive desirability: joy if the outcome is certain, hope if it is not certain. Fear is the result of knowing the source of the undesired event that might happen, while distress is the result of an event that already has happened or is going to happen for sure. If the source is unknown, the agent will feel anxiety. Anger is the result of an undesired event which has a responsible agent (notice it is possible to be angry with yourself), while guilt is the result of being the responsible agent for an event that is undesired in someone else's perspective. All variables always depend on a certain agent's perspective (p and q).

The total intensity for each emotion is calculated by taking the intensity for every single appraisal frame that has that emotion as *type* into account. The *intensity* for a single event is calculated by the formula

$$|desirability(p) \times likelihood(p)|$$

In this formula, *likelihood* is the probability of the event happening in the future. [Gra04].

Appraisal configuration	**Emotion**
$desirability(p) > 0$, $likelihood(p) = 1$	`joy`
$desirability(p) > 0$, $likelihood(p) < 1$	`hope`
$desirability(p) < 0$, $likelihood(p) = 1$	`distress`
$desirability(p) < 0$, $likelihood(p) < 1$	`fear`
$desirability(p) < 0$, $likelihood(p) < 1$, $state = NULL$	`anxiety`
$desirability(p) < 0$, $blameworthiness(q) = blameworthy$	`anger`
$desirability(q) < 0$, $blameworthiness(p) = blameworthy$, $p \neq q$	`guilt`

Table 4.4: Emotion categorization

4.1.2 Events

Each emotion is linked to one of the states in table 4.3 or table 4.2. These states are the result of events, which usually have a responsible person – all the possible values for the *responsible_agent* attribute are in table 4.5.

The value of the *blameworthiness* attribute declares whether the resposible agent should be blamed (`blameworthy`) or receive credit (`praiseworthy`) for the event. Finally, the attribute *status* states whether the event has already happened, `confirmed`, or has yet to happen, `unconfirmed`.

4.1.3 Structure

The WMEs in Soar can be graphically presented by a directed graph. To enhance in clarity, the structure of the locations of the data is presented as a tree; see figure 4.1.

The total values of each emotion are stored as in figure 4.2.

```
S1                              top state
+ appraisals
|  + types                      all possible types for emotion
|  |
|  + {Anger, Anxiety, ...}      type of this emotion
|  |  |
|  |  + emotion-type            type of this emotion
|  |  + importance              state importance
|  |  + intensity               of this emotion
|  |  + agent                   agent the appraisal is judged by
|  |  + appraisal               by which emotion was raised
|  |  |  |
|  |  |  + object
|  |  |  |  |
|  |  |  |  + name              the state
|  |  |  |
|  |  |  + type                 event helps or threatens state
|  |  |  + desired-self         binary mapping of importance
|  |  |  + evaluation
|  |  |  |  |
|  |  |  |  + responsible-agent  responsible agent for the event
|  |  |  |  + evaluation         whether responsible agent is
|  |  |  |                                  blamed or praised
|  |  |  + feature
|  |  |  |  |
|  |  |  |  + status             is event confirmed/unconfirmed
```

Figure 4.1: Tree structure of the locations of the emotion WMEs

Value	Person
`1sldr`	First squad (leader)
`2sldr`	Second squad (leader)
`3sldr`	Third squad (leader)
`4sldr`	Fourth squad (leader)
`ambulance`	The ambulance
`base`	The base
`lt`	Lieutenant (you)
`medevac`	The medevac (helicopter)
`medic`†	The team's medic
`mom`†	The mother
`sgt`†	The sergeant

Table 4.5: Possible values for *responsible_agent*; entries marked with a † are intelligent agents in the MRE system

4.2 Personality

Apart from the emotional state, an agent has a *personality*. This personality has influence on his coping strategy and (body) language. It can only be changed manually by the use of sliders, which is justified, as personality does not change in a short time interval as with a mission scenario.

```
S1                                  top state
+ io
|  |
|  + input-link                     input link
|  |  |
|  |  + {Defensiveness, ...}        type of personality with intensity as value
|  |  + self
|  |  |  |
|  |  |  + {Anger, Anxiety, ...}    total intensity of the emotion
|  |  |  + max                      emotion with the highest total intensity
```

Figure 4.2: Tree structure of the locations in the input link

The personality is defined by four *personality_types*: `defensiveness`, `terseness`, `initiative`, and `expressiveness`. As with the emotions, the intensity of a personality *personality_intensity* has a value between `0` and `1`. The location of these values are in the input link of the Soar-environment, see figure 4.2.

4.3 Speech acts

In this section possible utterances concerning emotions are evaluated. Some assumptions are made:

1. the speech acts are both domain and scenario independent
2. the emotional dialogue exists as a side line to existing dialogue
3. the emotional dialogue is always user driven

The first one is taken from section 3.2. The second assumption is made because, first of all, the current scenario isn't very suitable for having a natural emotional conversation. Also, blending emotional dialogue into existing dialogue in a natural way, would imply emotional, dialogue and social studies, which cannot be performed within the given time frame. The lack of dialogue context implies the third assumption, i.e. the emotional conversation is initiated by the user and has a question-answer construction. As mentioned in section 4.1, information about emotions are classified in the subjects *type*, *intensity*, *state*, $importance_{state}$, $desirability_{state}$, *likelihood*, *responsible_agent*, *blameworthiness*, $influence$, and *status*. Of these subjects, *type*, *intensity*, *state*, *responsible_agent* and *blameworthiness* can directly be used within conversations about emotions. We use this distinction for classifying the utterances in the following subsections.

The outline for a question-answer interaction consists of all possible questions with the same meaning, annotated with their keywords, and the possible answers for the questions, with the data that answer supplies and thus is needed to construct the answer.

Note: During implementation and testing certain revisions were made to the speech acts. In order to show the iteration process, the original speech acts are still depicted here. The final speech acts can be found in appendix A.

4.3.1 Emotion type

Our first question tries to find out how an agent feels. It's assumed his answer will give information about the emotion he feels the most, i.e. the emotion with the highest intensity. In addition to this information, the agent can answer more extensively, giving information on why he feels that way, how much he feels that way, et cetera.

Question How do you feel?

Keywords: *how, feel*

Question What's wrong with you?

Keywords: *what, wrong, you*

Answer I'm [emotion].

Data: *type*, *intensity*

Answer I'm a bit/really/very/... [emotion].

Data: *type*, *intensity*

Answer I'm a bit/really/very/... [emotion], because [additional information on state, responsible agent, ...]

Data: *type*, *intensity*, $influence$, *state*, *responsible_agent*, ...

The second question is simular to the first one, but now is focused on a particular state. Further attention should be given to the fact that this question could also imply a desire to get to know the agent's plan of action.

Question How do you feel about [state]?

Keywords: *feel, [state]*

Answer That doesn't seem really important at this time, sir.

Answer I'm [emotion] about [state].

Data: $type, intensity$

Answer I'm a bit/really/very/... [emotion] about [state].

Data: $type, intensity, state$

Answer I'm a bit/really/very/... [emotion] about [state], because [additional information on state, responsible agent, ...]

Data: $type, intensity, influence, state, responsible\_agent, \ldots$

The third question wants confirmation on a certain feeling. The answer consists of a confirmation and if positive, can be extended with information on why the agent feels the emotion. To be able to answer this question, a certain threshold needs to be set for the intensity.

Question Do you feel [emotion]?

Keywords: *feel, [emotion]*

Question Are you [emotion]?

Keywords: *are (to be), [emotion]*

Answer Yes / No, sir.

Data: $type, intensity$

Answer Yes, sir, because [additional information on state, responsible agent, ...]

Data: $type, intensity, influence, state, responsible\_agent, \ldots$

4.3.2 Intensity

A user should be able to reassure an agent, when the intensity of a certain emotion is too high. Although this isn't a real question, for clarity the same structure as in the other subsections is used.

The following utterances can be used for every emotion.

Question Calm down.

Keywords: *calm, down*

Question Relax.

Keywords: *relax*

Answer Yes, sir.

Data: $type, intensity$

Answer Will do, sir.

Data: $type, intensity$

Answer I am calm/relaxed, sir.

Data: $type, intensity$

Whether such an utterance of the lieutenant will have in fact an impact on the agent's emotion is a point of further attention.

4.3.3 State

If the user asks his interlocutor for the reasons of his emotional state, he's actually asking for the *state* an event helps or treatens, which as a result creates the agent's emotional state. In the answer a lot of information on this state has to be grouped.

Question Why are you [emotion]?

Keywords: *why, are (to be), [emotion]*

Question What's causing you to feel [emotion]?

Keywords: *what, causing, [emotion]*

Answer Because [state].

Data: $emotion, state, influence$

Answer It's [responsible agent]'s fault that [state]!

Data: $emotion, state, influence, responsible\_agent, blameworthiness$

Answer I'm not [emotion]!

Data: $emotion, intensity$

Answer That's personal, sir.

Data: $emotion$

4.3.4 Responsible agent and blameworthiness

The user might want to ask the interlocutor more details about the state, in particular who is responsible for the state and thus for the emotion. This question may depend on the emotion.

Question Who's responsible for [state]?

Keywords: *who, responsible, [state]*

Question Who's responsible for making you feel [emotion]?

Keywords: *who, responsible, [emotion]*

Question Towards whom do you feel guilty?

Data: $whom, guilty$

Answer [responsible agent].

Data: $emotion, state, influence, responsible\_agent, blameworthiness$

Answer It's [responsible agent]'s fault that [state]!

Data: $emotion, state, influence, responsible\_agent, blameworthiness$

4.4 Natural language

In chapter 2 the theories behind the natural language modules of MRE were presented. In a perfect world all speech would pass the speech recognizer, turning it into a string of words, which would be transformed into a set of semantic frames by the NLU module. These frames would travel through the system, updating the dialogue state, generating a communicative goal, selecting the proper content for output, and realizing the specific sentence which conveys that content. Unfortunately, we don't live in a perfect world. Real life isn't neat, it's filthy, dirty and full of hacks.

Although the 'neat' method described above is of course prefered over other ways of implementation, some problems concerning this method arise. First of all, the MRE project has currently no direct control over the core of the NLU module. This means that any natural language understanding added to MRE has to bypass this module, using Soar. Secondly, it is unclear if the NLG module can be used since there is no expert code maintainer at this moment.

This section discusses the various ways natural language understanding and natural language generation are implemented and how we can make use of them, bypassing both the NLU and part of the NLG.

4.4.1 Natural language understanding

The MRE system has two methods of bypassing the NLU module. One can either type in question propositions or one can define rules which scan for certain keywords in an utterance. Both these methods will create semantics which can continue the normal flow of the system.

The entered propositions should match the sequence `state-q object-id attribute value polarity`, where `state-q` serves for making clear a proposition is typed in, rather than natural language. The subject of the question should be indicated by typing `q` in it's place, i.e. the question "Are you feeling angry?" can be entered as `state-q sgt |Angry| yes q`, when adressing the sergeant. The typed in proposition is mapped to the proper semantics by placing the tokens in the `^semantics` attribute.

Keyword scanning is not actively used anymore, although some rules still exist. Figure 4.3 depicts a combination of these rules. It scans for the words "happened" and "here" in order to detect the utterance "What happened here?". When this rule fires, it generates the proper semantics by defining a question (`^type question`) with an event as subject (`^q-slot event`). The question is further defined as what event (`^type event`) happened at the Assembly Area (`^location aa`, which is where the scene takes place) in the past (`^time past`).

4.4.2 Natural language generation

The dialogue code will generate a certain obligation (also called *goal*) for the agent, for instance the obligation to assert a question. In order to adress this obligation, a communicative goal is formed, which holds the actual response. This is done by selecting one or more states (as discussed in 2.4.2) from all matching states. The system then generates natural language for the chosen alternative. The goal can be seen as why the agent is making an utterance, the comgoal is what this utterance is.

```
sp {top-state*apply*operator*understand-speech*nlu*what-happened-here
  (state <s> ^name top-state
             ^operator <o>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^id <id>
        ^speaker lt
        ^interpretation <i>)
  (<i> ^token.lex << happened >>
       ^token.lex << here >>)
-->
  (<i> ^addressee sgt
       ^mood question
       ^semantics <sem> + = >)
  (<sem> ^type question
         ^q-slot event
         ^prop <sem1>)
  (<sem1> ^type event
          ^location aa
          ^time past)}
```

Figure 4.3: Soar rule which maps the question "What happened here?" to the proper semantics

In terms of this assignment, there are three possibilities for generating language. The first method consists of defining new states, based upon the existing emotion code. This will make the system able to reason about emotions the same way it does in general. Two problems have to be adressed, though, when using this method. First, as an agent appraises all states in order to compute his emotions, adding emotion states (which then would also be appraised) could unbalance the system. Secondly, the system isn't capable of handling causality, so there's no way to get to know why an agent is, for example, feeling guilty. In order to make use of general reasoning rules, the system should be extended with causality features. The general outline of a proposition state can be found in figure 4.4.

The second method consists of directly mapping a goal to the proper semantics for the speech act. The NLG module then uses these semantics to generate certain sentences. This method bypasses the reasoning part of the system.

The third method bypasses both the content selection and sentence generating code, by mapping a certain goal directly to a speech act (`surface`). An example of such a rule can be found in figure 4.5, where a communicative goal of the sergeant answering the lieutenant whether he is hopeful is met by uttering "I'm hopeful". The fact that this rule will make the sergeant reply in this fashion even if he's in fact not hopeful, can be seen as a small indication that some work still remains to be done.

```
S1                          top state
+ current-state
|  |
|  + {state1, state2, ...}   state name
|  |  |
|  |  + attribute            subject of the state
|  |  + belief               whether agent thinks state is true or false
|  |  + concern              intrinsic (context free) value of state
|  |  + emotion              emotion of every agent for this state
|  |  + established-by       effect that caused current value of state
|  |  + id                   state name
|  |  + initial-belief       initial belief of the agent
|  |  + name                 state name
|  |  + negation             link to inverse state
|  |  + object-id            concerned agent
|  |  + polarity             positive or negative focused
|  |  + satisfied            is state satisfied (may differ from belief)
|  |  + sim-object           name of the object in the VR-simulation
|  |  + source               source of state (inference)
|  |  + type                 "state"
|  |  + value                value of the state
```

Figure 4.4: Tree structure of general propositions

```
sp {apply*output-speech*answer-emotion-type
  (state <s> ^agent-name sgt
             ^operator <o>)
  (<o> ^name output-speech
       ^comgoal <cg> )
  (<cg> ^speaker sgt
        ^addressee lt
        ^speech-act <b>)
  (<b> ^type backward
       ^action answer
       ^actor sgt
       ^question <sem>)
  (<sem> ^type question
         ^q-slot value
         ^prop <sem1>)
  (<sem1>  ^type value
           ^object-id sgt
           ^attribute |Hope|)
-->
  (<cg> ^surface |"I'm hopeful"| +  =)}
```

Figure 4.5: Tree structure of general propositions

Chapter 5

Design

This chapter describes the design of our system. Where possible, it uses methods already available.

We recall from subsection 4.4.1 that there are two alternatives for language understanding:

- Propositions
- Keyword scanning

As the use of propositions would imply abandoning the requirement of using natural language, this alternative is discarded and keyword scanning is chosen. For language generation three alternatives were presented:

- State reasoning
- Creating formal semantics manually
- Surfacing a string directly

Despite the problems mentioned in subsection 4.4.2 it showed that using state reasoning is possible. The MRE team allowed the addition of a `^cause` attribute to the state structre and implemented a generic follow-up why-question. It was asserted that possible issues with appraising emotion states could be dealt with. Also, it showed that using the NLG sentence planning ability could not be used within the given time frame, as it is quite domain specific. The design of the language generation therefore is a mixture between the first and third alternative. Emotion states will be created in order to make reasoning possible. When the system has come up with the communicative goal, though, the formal semantics will not be send to the NLG sentence generation. Rather, the semantics form a trigger to use certain templates.

As states form the informational basis for both the reasoning part and the creation of formal semantics, this subject will be discussed first in section 5.1. Section 5.2 uses the states as a starting point to design the natural language understanding part. In section 5.3 the required templates are designed. Section 5.4 concludes with an overview of the design.

Certain issues during later phases of the assignment caused some changes to the original design. In order to show some insight in the process, the original design is shown in the following sections, revisions can be found in section 5.5.

5.1 Emotion states

Making the required data for the output available is done by creating states as mentioned above. The current state design as depicted in figure 4.4 is used, with the addition of a `cause` attribute. These new states are called *emotion states*. The states contain the data that is *in* the speech act; additional data (i.e. terseness of the agent to determine the template) is extracted from the rest of the system. The actual template fillings will be placed in lookup tables – see subsection 5.3.2 for more details on this subject.

Since it is not desirable to have states for every single subject that can be talked about, certain states are created *on-the-fly*. For example, if the user asks for the cause of a certain feeling, the state containing that information is only available as long as the question is under discussion, otherwise every reason for every feeling should be available in permanent states. Some states are *permenant*, though, since there is only one piece of information needed, i.e. the answer to the question "How do you feel?" only requires a single state of information.

The following subsections describe every emotion state related to a question. Following the original design of the states, every state has a negated state ($belief$ = `false` instead of the $belief$ = `true`); only the states with $belief$ = `true` are listed for each question. The value for $initialbelief$ must be equal to that of $belief$, because of a syntax requirement of the system.

The values for certain emotion states depend on what the intensity of the emotion is. If an agent feels a certain emotion with an intensity less than 0.2, he does not believe he feels the emotion. This threshold has effect on the creation of emotion states, but also on the selection of the template alternative and the template fillings.

Note: In the figures in these sections `plain text` defines an attribute value. A text between `<` and `>` defines a value that has to be looked up using an algorithm. Text between `[` and `]` defines a value that can be acquired from the user's speech act, and text preceded by a `*` define a path of attributes to a value in the system.

5.1.1 Emotion type

When the question "How do you feel?" is uttered, the adressee is asked for his *most important* feeling, so the emotion state attribute will be `max-feeling`. The value of the state is the most important emotion, which is the emotion the agent feels most and can be found in `s1.io.input-link.self.max`. As for every emotion state the value in `s1.agent-name` applies for the $object - id$ attribute. Since the most important feeling always exists, *polarity* is `positive`. The *cause* attribute links to the object of the most important state generating this feeling, the state that has the greatest part in boosting the most important emotion – a link to the appraisal object of this state is provided in the *source* attribute.

This emotion state will be available permanently.

How do you feel?	
attribute	`max-feeling`
belief	`true`
cause	`<most important state for this emotion>`
initial − belief	`true`
object − id	`*s1.agent-name`
polarity	`positive`
satisfied	`true`
source	`<appraisal for most important state>`
type	`state`
value	`*s1.io.input-link.self.max`

The emotion state for a question as "How do you feel about [state]?" will be created on-the-fly. The value for the answer can be found in the appraisal of the mentioned state.

How do you feel about [state]?	
attribute	`feeling-about-state`
belief	`true`
cause	`*current-state.[state]`
initial − belief	`true`
object − id	`*s1.agent-name`
polarity	`positive`
satisfied	`true`
sim − object	`*current-state.[state].sim-object`
source	`<appraisal for state>`
type	`state`
value	`*current-state.[state].[emotion]?`

For every emotion an emotion state `feeling` is created. The value of *polarity* states whether the agent believes he feels this emotion or not; this depends on the threshold for the intensity. If the agent does feel the emotion, causality is added in the *cause* attribute and the appraisal source is added in the *source* attribute.

Do you feel [emotion]?	
attribute	`feeling`
belief	`true`
cause	`<most important state for this emotion>` if $intensity_{[emotion]} > 0.2$
initial − belief	`true`
object − id	`*s1.agent-name`
polarity	`positive`, if $intensity_{[emotion]} > 0.2$ `negative`, if $intensity_{[emotion]} \leq 0.2$
satisfied	`true`
sim − object	
source	`<appraisal for most important state>` if $intensity_{[emotion]} > 0.2$
type	`state`
value	`*s1.io.input-link.self.[emotion]`

5.1.2 Intensity

For the reactions on an utterance as "Calm down." the emotion states `max-feeling`, as described above, are used.

5.1.3 State

For the reaction on a question "Why are you [emotion]?" the value of *polarity* and *cause* in the `feeling` state attribute for that specific emotion is used, so no new emotion states need to be designed here.

5.1.4 Responsible agent and blameworthiness

For the answer to the question "Who's responsible for making you feel [emotion]?" new (permanent) emotion states are needed. For every emotion an emotion state `responsible-for-[emotion]` is created only if the intensity of that emotion is higher than the threshold of 0.2. The *value* is the agent that is responsible for the state that influences the emotion the most. In the *cause* attribute, the blameworthiness of the responsible agent is stored. Note that in this case the *cause* attribute is not a state.

Who's responsible for [state]?	
attribute	`responsible-for-[emotion]`
belief	`true`
cause	`<blameworthiness>`
initial − belief	`true`
object − id	`*s1.agent-name`
polarity	`positive`
satisfied	`true`
type	`state`
value	`<responsible agent for most important state>`

5.2 Natural language understanding

With bypassing the NLU module cannot be used, semantic frames have to be created by Soar, given a certain utterance. More specific this means *scanning* the input for specific *keywords*, and on detection of the right keywords generating the proper semantics, which continue the normal flow within the system, i.e. dialogue and language generation code.

In the following subsections the semantics for each type of question are presented, accompanied by the keywords mentioned in section 4.3. The first colomn of a table shows the natural language question, with the variable `emotion` or `state` between straight brackets. The second column shows the keywords needed in order to detect the question. All the words connected with $\wedge$ are needed for detection; $\vee$ denotes a choice between certain words. The semantics associated with the question(s) are shown in the third column, based upon the states discussed in section 5.1.

The last subsection discusses the mapping from emotions and states in natural language to the representation used by the system.

5.2.1 Emotion type

The semantics for the emotion type questions are straightforward: the first and second are about the value of respectively the states `max-feeling` and

`feeling-about-state`. As there's only one `max-feeling` no additional information is needed. In order to get the right `feeling-about-state`, the `cause` attribute is set to the state asked about.

The third question is about whether a certain state is true or false, so the `q-slot` is set to polarity. The value of the `value` attribute will make sure the right `feeling` state is selected.

Question	Keywords	Semantics
How do you feel?	how ∧ feel	`^type question ^q-slot value`
What's wrong with you?	(what ∨ what's) ∧ wrong ∧ you	`^attribute max-feeling`
How do you feel about [state]	feel ∧ [state]	`^type question ^q-slot value ^attribute feeling-about-state ^cause <state>`
Do you feel [emotion]?	feel ∧ [emotion]	`^type question ^q-slot polarity`
Are you [emotion]?	are ∧ [emotion]	`^attribute feeling ^value <emotion>`

5.2.2 Intensity

Since this type of question is more or less a gimmick, no states have been designed. In order to detect this question, a dummy is used to create fake semantics. Although the formal semantics do not match the informal semantics, it is possible to detect the question later on and present the proper reply.

Question	Keywords	Semantics
Calm down.	calm ∧ down	`^type question ^q-slot dummy-int`
Relax.	relax	`^attribute max-feeling`

5.2.3 State

When one asks "Why do you feel angry?", we expect to get the cause of the emotion as an answer. This semantics is presented formally in the table below.

Question	Keywords	Semantics
Why are you [emotion]?	why ∧ are ∧ [emotion]	`^type question ^q-slot cause`
What's causing you to feel [emotion]	(what ∨ what's) ∧ causing ∧ [emotion]	`^attribute feeling ^value <emotion>`

5.2.4 Responsible agent and blameworthiness

Both of the quesions below are semanticly the same, but phrased differently. Since a responsibility state exists for every emotion, it is merely a matter of asking for the value of that state.

Question	Keywords	Semantics
Who's responsible for making you feel [emotion]?	(who ∨ who's) ∧ responsible ∧ [emotion]	`^type question ^q-slot value ^attribute responsible-for-[emotion]`
Towards whom do you feel guilty?	(whom ∨ who) ∧ guilty	

5.2.5 Synonyms

As the emotions and states are internally represented in a certain way, we need to map natural language words to these internal representations. This also provides us with the ability to use synonyms.

For this mapping we make additional look-up tables. The emotion synonyms can be seen in the table below. Although in natural language not all synonyms map exactly to one emotion or state, we use this many-to-one mapping for sake of simplicity. This will result in the agent interpreting a user's ambiguous question (like "Do you feel stressed?") in a way the user might not have intended[1].

Natural language	**Internal representation**
anger, angry, annoyed, irritated	`|Anger|`
anxiety, anxious, worried, concerned	`|Anxiety|`
distress, distressed, upset, disturbed, troubled	`|Distress|`
fear, fearful, afraid, frightened, stressed	`|Fear|`
guilt, guilty	`|Guilt|`
hope, hopeful, optimistic	`|Hope|`
joy, joyful, exited, happy	`|Joy|`

In order to detect the state the user asks about (for instance `boy-healthy`), a lookup table with boolean functions is used (table 5.1). In order to make general questions like "How do you feel about the boy?" possible, questions which contain only one keyword will be directed to one of the most likely states the user is interested in.

5.3 Natural language generation

The agents' answers will be constructed by the use of *templates*, as explained in section 5.3.1. The information needed to fill in the templates must be provided by the system; this is explained in more detail in section 5.3.2.

5.3.1 Templates

When the NLG module is not used for creating sentences, either fixed strings (*canned text*) or *templates* are alternatives. Since templates are more flexible and dynamical [Jur00], this method is used mainly for constructing the agents' utterances.

For each of the questions mentioned in section 4.3 a template type is available; each type has one or more alternatives, depending for instance on the terseness of the agent. The template types are listed in the following subsections. In what way the templates are filled is described in section 5.3.2.

The template type is selected in Soar, by detecting assertions to certain types of questions, as discussed in section 5.2. Once a template type is selected, a TCL script will be called, which selects and fills in a certain alternative, returning the string to Soar.

EM is defined as the set containing the emotions: $EM = \{\texttt{anger}, \texttt{anxiety}, \texttt{distress}, \texttt{fear}, \texttt{guilt}, \texttt{hope}, \texttt{joy}\}$.

[1] One can argue whether we are just lazy not implementing a clarification system, or whether this one-to-many method makes the agent more human-like. We tend to favor the latter.

Natural language	**Internal representation**
crowd ∧ goodwil	`maintain-goodwill`
crowd ∧ anger	`crowd-angry`
(boy ∨ child ∨ kid) ∧ dead	`boy-dead`
boy ∨ child ∨ kid	`boy-healthy`
(boy ∨ child ∨ kid) ∧ minor ∧ injuries	`minor-injuries`
(boy ∨ child ∨ kid) ∧ serious ∧ injuries	`serious-injuries`
(boy ∨ child ∨ kid) ∧ critical ∧ injuries	`critical-injuries`
driver	`driver-healthy`
driver ∧ (injuries ∨ injured)	`driver-minor-inj`
mom ∨ mother ∨ woman	`mother-healthy`
(boy ∨ child ∨ kid) ∧ health ∧ known	`know-boy-health`
(squads ∨ sqds) ∧ transit	`sqds-in-transit`
(lieutenant ∨ lt) ∧ transit	`lt-in-transit`
(lieutenant ∨ lt) ∧ (aa ∨ (assembly ∧ area))	`lt-at-aa`
(lieutenant ∨ lt) ∧ celic	`lt-at-celic`
(sergeant ∨ sgt) ∧ (aa ∨ (assembly ∧ area))	`sgt-at-aa`
medic ∧ (aa ∨ (assembly ∧ area))	`medic-at-aa`
(boy ∨ child ∨ kid) ∧ (aa ∨ assembly ∧ area)	`at-boy-aa`
((1st ∨ first) ∧ (sqd ∨ squad)) ∧ (aa ∨ (assembly ∧ area))	`1st-sqd-at-aa`
((1st ∨ first) ∧ (sqd ∨ squad)) ∧ celic	`1st-sqd-at-celic`
((1st ∨ first) ∧ (sqd ∨ squad)) ∧ (lz ∨ (landing ∧ zone))	`1st-sqd-at-lz`
same for 2nd, 3rd and 4th squad	
((1st ∨ first) ∧ (sqd ∨ squad)) ∧ transit	`1st-sqd-in-transit`
((4th ∨ fourth) ∧ (sqd ∨ squad)) ∧ transit	`4th-sqd-in-transit`
(boy ∨ child ∨ kid) ∧ hospital	`at-boy-hospital`
(mom ∨ mother ∨ woman) ∧ intersection	`mom-in-intersect`
(sqd ∨ squad) ∧ intersection	`sqd-in-intersect`
medevac ∧ (aa ∨ (assembly ∧ area))	`medevac-at-aa`
medevac ∧ base	`medevac-at-base`
medevac	`medevac-called`
(medevac ∧ overhead)	`medevac-overhead`
amb ∧ (aa ∨ (assembly ∧ area)	`amb-at-aa`
amb ∧ base	`amb-at-base`
amb	`ambulance-called`
route ∧ (secure ∨ secured)	`secure-route`
(aa ∨ (assembly ∧ area)) ∧ (secure ∨ secured)	`aa-secure`
(accident ∨ site) ∧ (secure ∨ secured)	`accident-secure`
(12 ∨ twelve) ∧ (4 ∨ four) ∧ (secure ∨ secured)	`12-to-4-secure`
(4 ∨ four) ∧ (8 ∨ eight) ∧ (secure ∨ secured)	`4-to-8-secure`
(8 ∨ eight) ∧ (12 ∨ twelve) ∧ (secure ∨ secured)	`8-to-12-secure`
(lz ∨ (landing ∧ zone)) ∧ (secure ∨ secured)	`lz-secure`
(lz ∨ (landing ∧ zone)) ∧ (mark ∨ marked)	`lz-marked`
(lz ∨ (landing ∧ zone)) ∧ (clear ∨ clearedd ∨ free)	`lz-clear`
(((2 ∨ two) ∧ (6 ∨ six)) ∨ unit ∨ team) ∧ (support ∨ supporting)	`support-1-6`
(((2 ∨ two) ∧ (6 ∨ six)) ∨ unit ∨ team) ∧ together	`retain-mass`
(((2 ∨ two) ∧ (6 ∨ six)) ∨ unit ∨ team) ∧ (fracture ∨ fractured ∨ split)	`fracture-unit`
((1 ∨ one) ∧ (6 ∨ six)) ∧ celic	`1-6-at-celic`
hospital ∧ tuzla	`hospital-at-tuzla`

Table 5.1: Synonyms for states

As noted, the template fillings will be discussed in section 5.3.2.

Emotion type

As an answer to the question how one feels, i.e. "How do you feel?", four template alternatives are available. What emotion *emotion* is returned in the answer depends on the intensity of the emotions:

$$emotion \in EM, \forall e \in EM : intensity_{emotion} \geq intensity_e$$

The first template alternative is in effect if the maximum emotion has an intensity small enough to make that agent belief he doesn't feel it (compare with the answer to "Do you feel [emotion]?"). If the intensity is high enough, the chosen template alternative depends on the terseness $personality\_intensity_{terseness}$ of the agent:

1. I'm fine, sir.
   $intensity_{emotion} \leq 0.2$
2. I feel [emotion], sir.
   $intensity_{emotion} > 0.2 \wedge personality\_intensity_{terseness} > 0.75$
3. I feel [intensity] [emotion], sir.
   $intensity_{emotion} > 0.2 \wedge 0.25 < personality\_intensity_{terseness} \leq 0.75$
4. I feel [intensity] [emotion], because of [state-pre] [influence] [status] [state-post], sir.
   $intensity_{emotion} > 0.2 \wedge personality\_intensity_{terseness} \leq 0.25$

To the question how one feels about a certain state, two different template alternatives are available. Here, too, does the chosen alternative depend on the terseness of the agent:

1. I feel [emotion] about it, sir.
   $personality\_intensity_{terseness} > 0.5$
2. I feel [intensity] [emotion] about it, sir.
   $personality\_intensity_{terseness} \leq 0.5$

The third question was about confirmation on a certain emotion. The given answer depends on the intensity of the emotion asked for and the terseness of the agent:

1. Not at all, sir.
   $intensity_{emotion} \leq 0.1$
2. No, sir.
   $0.1 < intensity_{emotion} \leq 0.2$
3. I don't, sir.
   $0.1 < intensity_{emotion} \leq 0.2$
4. Yes, sir.
   $0.2 < intensity_{emotion} \leq 0.5 \wedge personality\_intensity_{terseness} > 0.25$

5. I do, sir.
   $0.2 < intensity_{emotion} \leq 0.5 \wedge personality\_intensity_{terseness} > 0.25$

6. Certainly, sir.
   $intensity_{emotion} > 0.5 \wedge personality\_intensity_{terseness} > 0.25$

7. I do, sir, because of [state-pre] [influence] [status] [state-post], sir.
   $intensity_{emotion} > 0.2 \wedge personality\_intensity_{terseness} \leq 0.25$

Intensity

In contradiction to all other speech acts, the speech acts concerning the intensity ("Calm down.") do not have a question-answer construction, but the user is trying to soothe the agent. The agent's reaction depends on the intensity of all emotions, i.e. for every emotion the intensity needs to be really low to give a negative reaction:

1. Yes, sir.
   $\exists emotion \in EM : intensity_{emotion} > 0.20$

2. Will do, sir.
   $\exists emotion \in EM : intensity_{emotion} > 0.20$

3. I am calm, sir.
   $\forall emotion \in EM : intensity_{emotion} \leq 0.20$

4. I am already relaxed, sir.
   $\forall emotion \in EM : intensity_{emotion} \leq 0.20$

State

The reason why an agent feels a certain way has already been discussed before, in the most expressive version of the question "How do you feel?". When explicitly asked for the reason of a certain feeling, the agent may either answer that question by naming a certain state or rejecting that question, depending on the intensity of the emotion *emotion* asked for.

Note that it is only possible to explicitly blame one ("Your fault!") if someone is actually to be blamed ($blameworthiness = \texttt{blameworthy}$).

1. I'm not feeling [emotion], sir.
   $intensity_{emotion} \leq 0.20$

2. That's personal, sir.
   $intensity_{emotion} > 0.20 \wedge personality\_intensity_{defensiveness} > 0.75$

3. Because of [state-pre] [influence] [status] [state-post], sir.
   $intensity_{emotion} > 0.20 \wedge (\neg\exists r : responsible\_agent_{emotion} = r \vee 0.25 < personality\_intensity_{defensiveness} \leq 0.75 \vee (personality\_intensity_{terseness} \leq 0.75 \wedge (blameworthiness = \texttt{praiseworthy} \vee emotion = \texttt{hope} \vee emotion = \texttt{joy})))$

4. The [state-pre] [influence] [status] [state-post], sir, that's [responsible_agent_poss] fault!
   $intensity_{emotion} > 0.20 \wedge \exists r : responsible\_agent_{emotion} = r \wedge$
   $personality\_intensity_{defensiveness} \leq 0.25 \wedge blameworthiness = \texttt{blameworthy} \wedge$
   $\neg(emotion = \texttt{hope} \vee emotion = \texttt{joy})$

Responsible agent and blameworthiness

Although the question "Who's responsible for [state]?" can be a logical part of an emotion dialogue, it does not directly touch on the subject of emotion. For the sake of time, this question therefore is left for the original MRE team to implement.

The subject of the responsible agent has already been discussed above, but there's a small difference in the answer to the question "Who's responsible for making you feel [emotion]?". The agent replies by naming the responsible agent, which makes it not sound to use the same template fillings as in the section above, because a pronoun is needed instead of a possessive pronoun.

It depends on the intensity of the emotion the responsible agent is involved in whether the responsible agent is mentioned in the answer. If the intensity of the emotion is less than the threshold, the agent does not actually consider the emotion, so he replies negatively.

Since not all states have to have a responsible agent, if the intensity is greater than the threshold, first it is checked if the responsible agent exists. If it does not exist, the agent needs to reply accordingly.

1. I'm not [emotion], sir.
   $intensity_{emotion} \leq 0.20$

2. I don't know who is responsible, sir.
   $intensity_{emotion} > 0.20 \wedge \neg\exists r : responsible\_agent_{emotion} = r$

3. [Responsible_agent], sir.
   $intensity_{emotion} > 0.20 \wedge \exists r : responsible\_agent_{emotion} = r \wedge$
   $(personality\_intensity_{terseness} > 0.25 \vee blameworthiness = \texttt{praiseworthy} \vee$
   $emotion = \texttt{hope} \vee emotion = \texttt{joy})$

4. That's [responsible_agent_poss] fault, sir!
   $intensity_{emotion} > 0.20 \wedge \exists r : responsible\_agent_{emotion} = r \wedge$
   $personality\_intensity_{terseness} \leq 0.75 \wedge blameworthiness = \texttt{blameworthy} \wedge$
   $emotion \neq \texttt{hope} \wedge emotion \neq \texttt{joy})$

5.3.2 Pieces of objects

To be able to fill in the templates, the information provided by the system has to be converted into usable pieces. The information deducted from the relevant objects will be converted to natural language strings and stored in lookup tables in the top state. In this subsection all the values for the different attributes are presented. At this moment only one possible value is given. Because Soar has the possibility of randomly picking an entry if more attributes with the same name exist, it is possible to increase the number of translations, making the answers of the agents more diverse and enhancing the use of natural language.

Emotions

Emotion	Natural language string
`anger`	angry
`anxiety`	anxious
`distress`	distressed
`fear`	fearful
`guilt`	guilty
`hope`	hopeful
`joy`	joyful

Table 5.2: Values for the `nl-emotion` attribute

The individual emotions are used in several templates. The translation to natural language for the most important emotion is stored in the attribute `^nl-emotion`. The translations are listed in table 5.2.

Intensity

Intensity i	Natural language string
$i \leq$ `0.25`	a little
`0.25` $< i \leq$ `0.50`	pretty
`0.50` $< i \leq$ `0.75`	really
`0.75` $< i$	very

Table 5.3: Values for the `nl-intensity` attribute

The intensity of the most important emotion is stored in the attribute `^nl-intensity` in the top state. Translations are in table 5.3.

State

The translation of the states to natural language is split into two pieces, namely `^nl-state-pre` and `^nl-state-post`. This is necessary, since some templates require certain words to be inserted between these two parts, like the verb in a sentence and optionally a negational word ("not"). Both pieces of text for every status are listed in table 5.8 on page 40.

Since it is only possible to have a conversation in English with the sergeant and the medic, only the states for these two agents are transformed to natural language.

Influence

Influence	Natural language string
`facilitator`	<nothing>
`inhabitor`	not

Table 5.4: Values for the `nl-influence` attribute

The type attribute of an appraisal (table 5.4) indicates whether the event facilitates (helps) or inhibits (threatens) the appraised state. This information is used to express if the emotion is about progress towards or the blocking of a (un)desired state. In the second case, the word "not" is added, see table 5.4.

Status

Status	Natural language string
`confirmed`	being
`unconfirmed`	going to be

Table 5.5: Values for the `nl-status` attribute

For every state, the attribute status annotates whether the state has already happened (`confirmed`) or if it not yet has happened (`unconfirmed`). If the state has not yet happened, the feelings of the agent are about the future, so the verb of the output sentence should by altered to future tense in that case, as shown in table 5.5.

Responsible agent

Responsible agent	Natural language string
`1sldr`	The first squad leader / Johnson
`2sldr`	The second squad leader
`3sldr`	The third squad leader
`4sldr`	The fourth squad leader / Lopez
`ambulance`	The ambulance
`base`	The base
`lt`	You are
`medevac`	The medevac
`medic`†	The medic / Tucci
`mom`†	The mother
`sgt`†	The sergeant

Table 5.6: Values for the `nl-responsible_agent` attribute

Every state has a certain responsible agent and it's possible the agent wants to express this information in his utterance, especially when he feels angry about someone messing something up. Both the pronoun `nl-responsible_agent` and the possessive pronoun `nl-responsible_agent_poss` of every agent are needed (see subsection 5.3.1). The different values for the responsible agents are in table 5.6 and table 5.7. These values have one exception: for the intelligent agents in the system (marked with a †) respectively "my" and "I am" applies for the value that is equal to themselves, i.e. for the sergeant, the value of `sgt` will be "my" and "I am" instead of "the sergeant's" and "The sergeant".

Responsible agent	Natural language string
`1sldr`	first squad leader's / Johnson's
`2sldr`	second squad leader's
`3sldr`	third squad leader's
`4sldr`	fourth squad leader's / Lopez's
`ambulance`	the ambulance's
`base`	the base's
`lt`	your
`medevac`	the medevac's
`medic`†	the medic's / Tucci's
`mom`†	the mother's
`sgt`†	the sergeant's

Table 5.7: Values for the `nl-responsible_agent_poss` attribute

5.4 Overview

An overview of the design is depicted in figure 5.1. The straight path (indicated by bold lines) is normally taken with fully implemented questions and answers. We will bypass both the NLU module and part of the NLG module, as described earlier

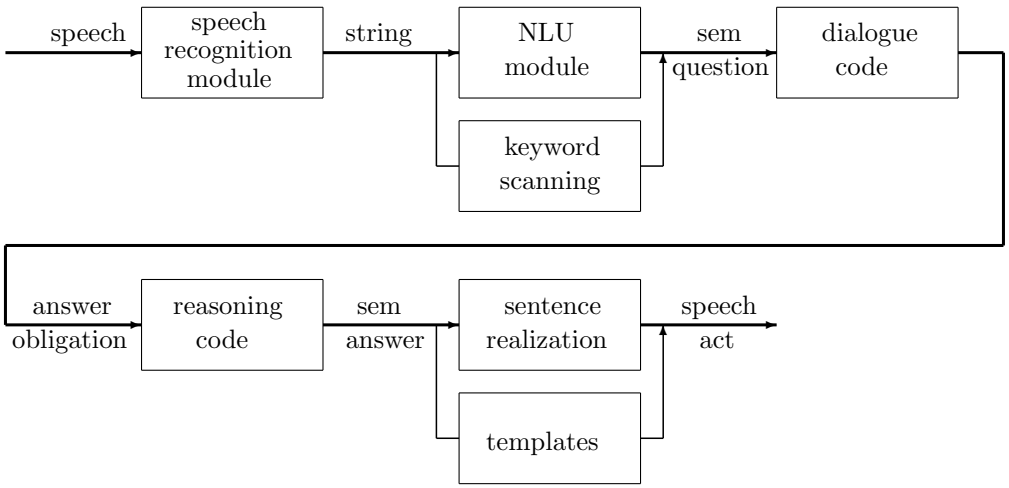

Figure 5.1: Dataflow user-agent emotional dialoge

5.5 Iteration

This section will discuss the encountered aspects during later phases, which led to changes to the original design.

State	nl-state-pre	nl-state-post
maintain-goodwill	the goodwill of the crowd	maintained
crowd-angry	the crowd	angry
boy-dead	the boy	dead
boy-healthy	the boy	healthy
minor-injuries	the boy	slightly injured
serious-injuries	the boy	seriously injured
critical-injuries	the boy	critically injured
driver-healthy	the driver	healthy
driver-minor-inj	the driver	slightly injured
mom-healthy	the mother	healthy
know-boy-health	the health of the boy	known
sqds-in-transit	the squads	in transit to the assembly area
lt-in-transit	the lieutenant	in transit to the assembly area
lt-at-aa	the lieutenant	at the assembly area
lt-at-celic	the lieutenant	at Celic
sgt-at-aa	the sergeant	at the assembly area
medic-at-aa	the medic	at the assembly area
at-boy-aa	the boy	at the assembly area
mom-at-aa	the mother	at the assembly area
1st-sqd-at-aa	the first squad	at the assembly area
1st-sqd-at-celic	the first squad	at Celic
1st-sqd-at-lz	the first squad	at the landing zone
	(same for 2nd, 3rd and 4th)	
1st-sqd-in-transit	the first squad	in transit to the assembly area
4th-sqd-in-transit	the fourth squad	in transit to the assembly area
at-boy-hospital	the boy	at the hospital
mom-in-intersect	the mother	at the intersection
sqd-in-intersect	the squad	at the intersection
medevac-at-aa	the medevac	at the assembly area
medevac-at-base	the medevac	at the base
medevac-called	the medevac	called for
medevac-overhead	the medevac	overhead
amb-at-aa	the ambulance	at the assembly area
amb-at-base	the ambulance	at the base
ambulance-called	the ambulance	called for
secure-route	the route to Celic	secured
aa-secure	the assembly area	secured
accident-secure	the accident site	secured
12-to-4-secure	the assembly area	partly secured
4-to-8-secure	the assembly area	partly secured
8-to-12-secure	the assembly area	partly secured
lz-secure	the landing zone	secured
lz-marked	the landing zone	marked by green smoke
lz-clear	the landing zone	cleared of civilians
support-1-6	our team	active supporting Eagle 2-6
retain-mass	our team	together
fracture-unit	our team	split up
1-6-at-celic	our team	at Celic
hospital-at-tuzla	the hospital	at Tuzla

Table 5.8: Values for the `nl-state-pre` and the `nl-state-post` attributes

5.5.1 Keyword scanning

As noted in chapter 4.3, certain changes were made to the speech acts. A list of speech acts which were implemented can be found in appendix A. Some of these speech acts were not detected properly when using the keywords in section 5.2. As a result, different keywords are used in the actual implementation and the absence of keywords is also checked. The Soar code of `language-emia.soar` in appendix B shows the implemented keywords.

During the design we weren't aware of the fact that a lexicon already existed. This lexicon already contained most of the synonyms needed, so it was extended with some additional entries, like the emotion synonyms. The boolean functions for detecting states caused some problems, as these are hard to implement in a static Soar look-up table. This will be discussed in section 6.3.

During the implementation it became clear that the formal semantics didn't lead to the expected results if the `type` attribute within the properties of the semantics wasn't set. Also, we learned that although not really necessary for our purposes, it is good practise to include an object-id. Therefore all semantics were extended with `^type state` and `^object <self>`.

It is noted that the use of the state attributes, like `max-feeling`, as the value for the semantic `^attribute`, will in most cases restrict the matching states for answering a question to one. A more sophisticated mechanism could lead to more complicated answers. More on this subject in the next chapter.

5.5.2 Templates

During implementation, it became clear that the design of the templates did not fulfill our demands in terms of how natural their output was. Even though they looked good on paper, in discussion with the agents the sentences seemed artificial. To enhance the quality of the sentences, the following changes have been made:

1. The values for the *status* and the $influence$ attribute are linked together to choose an output utterance, instead of them individually selecting template fillings. The design in table 5.9 is now used, instead of the one in tables 5.4 and 5.5.

2. The templates that included causality have been modified: the part of sentences that read "... because of [state-pre] [influence] [status] [state-post] ..." have been changed to "... because [state-pre] [influence-status] [state-post] ...", [influence-status] being the value found in table 5.9.

Influence	**Status**	**Natural language string**
`facilitator`	`confirmed`	is
`inhabitor`	`confirmed`	is not
`facilitator`	`unconfirmed`	will probably be
`inhabitor`	`unconfirmed`	probably won't be

Table 5.9: Values for the `nl-influence-status` attribute

Furthermore, some changes were made in the way an utterance is exactly phrased. For instance, the agent now answers a question about the boy or driver

with “I’m worried about him.”, rather than “I’m worried about it.”. In some cases these alterations lead to incorrect English, but we’ve chosen to give priority to natural and correct language for most cases and small discrepancies for others over correct but innatural language for all cases. All types of speech acts can be seen in appendix A, the exact implementation can be seen in appendix B.

Chapter 6

Implementation

6.1 Introduction

This chapter shows how the design in the last chapter was implemented, using Soar and TCL. It discusses aspects of both implementation and testing which led to problems or certain insights. The first section is about emotion states. Natural language understanding and generation will be discussed in subsequent sections. All the Soar and TCL code can be found in appendix B.

6.2 Emotion states

The emotion state `max-feeling` is created, using the `^max` attribute of the input-link, which can be found in `io.input-link` on the top state. The seven `feeling` states are created by taking the aggregate intensity out of the input-link and letting the polarity depend on this intensity and the designed threshold. For every `responsible-for-[emotion]` (e.g. `responsible-for-anger` and `responsible-for-hope`) an individual rule is implemented, which depends in the left-hand-side on the existence of a `feeling` emotion state with positive polarity: if the agent does not feel a certain emotion, there cannot be a responsible agent for that emotion. Finally, the on-the-fly `feeling-about-state` emotion state is elaborated if the right operator is in effect (*output-speech*) and there actually exists a significant appraisal about that state (significance is implemented by a small threshold on the intensity of an appraisal). If one or more significant appraisals do exist, the appraisal with the highest intensity is chosen and the emotion of that appraisal is taken as the value for the `feeling-about-state` state.

Causality for `max-feeling` and `feeling` emotion states is added after the elaboration of the states. For every emotion, the appraisal objects are scanned for the object with the highest intensity. A link to the appraisal object is saved in the `^source` attribute, while the appraised state is saved in the `cause` attribute.

One last rule adds the simulation object to the `feeling-about-state` emotion state. A simulation object is the name of one of the objects in the virtual world, which for example can be used to have agents gaze at that object. Unfortunately, the names of these object do not equal the names in the Soar system, so they have to be added explicitly.

6.3 Natural language understanding

The basis of the natural language understanding part is formed by a type of Soar rule that maps a certain question related to emotions to the right semantics. For the left-hand side of this rule, two things are important. First, how to detect the right sentences, and second, how to map natural language to the the formal representation of emotions and states. For detecting the right sentences, keywords are used as described in section 5.2. In order to avoid ambiguity we use both words that must be present and those that must be absent. This makes it possible to distinct between "How do you feel?" and "How do you feel about [state]?" when the question "How do you feel about the boy being injured?" is uttered.

For mapping natural language to the formal representation both the existing lexicon and new code is used. The lexicon is expanded with emotions so, for instance, both "angry" and "anger" are mapped to the internal representation `|Anger|`. The implementation of state detection caused a bit of a problem, as boolean functions like in table 5.8 are hard to represent in a static Soar structure. A Soar rule for every state would be a solution, mapping certain words directly to the appropiate state, but this is of course very labour extensive, hard to maintain and also not very neat. The existing lexicon is therefore extended with a list of all the states and all the words which map to a certain state. Figure 6.2 shows part of this list. The problem now is that one single word already leads to a state and in cases like "boy" to multiple states. Therefore three Soar rules were made, which would scan for either one, two or three keywords which could lead to a state. When all posibilities are found, a Soar rule will select the one with the highest priority, i.e. the most keywords (the most detailed description). The selection of which keywords are needed for a certain state is now somewhat restricted, as in no situation a lower priority match is to be better than than a higher priority match. Originally, when more than one candidate with the same priority was present, the system would randomly choose one. During testing it was noticed, that this often resulted in unnatural situations where the agent dismissed the question as not being important, while it cleary was. This is particulary true for the question "How do you feel about the boy?" which leads to eight possible states, of which only two are relevant. Now, all the states that don't lead to an emotion with a certain intensity are discarded and a random choice between all remaining candidates is made. The randomization should make conversation more dynamic, as opposed to always selecting the one with the highest assiociated emotion. As there is already a threshold implemented for linking emotions to states, these random answers should be sensible. It is noted that this selection mechanism is performed on the understanding side, rather than the generation side. More about this at the end of this section.

As Soar computes all possibilities of how certain keywords lead to states, it can be rather processor intensive. When a question contains three keywords, for instance, it not only finds the states which can be reached with these three words, but also all possiblities of how to reach states with one and two words. For now the impact doesn't seem to be too big, but if it turns out that these rules take relatively too much processor time, one could alter the mechanism a bit by putting a counter on the top state which holds track of the number of keywords needed in order to find the right state. It can be set to three at first in order to find all states for three found keywords. If no results show, it will

be set to two, so all results with two keywords can be found. This will ensure that no time is wasted on finding states with a lower priority then necessary.

The right-hand side of the basic Soar rule is a rather straight forward implementation of the design given in section 5.2, with a few exceptions. First of all, a priority attribute and an attribute `emotion-semantics` were added. This provides an opportunity to gather all possible semantics for the emotion question and select an appropiate one. After that, the emotion semantics is given a higher preference than the semantics found by the NLU, as this for now delivers only garbage. In order to get the right state name when asking about responsiblity (for instance `responsible-for-anger`) a TCL rule is used, as Soar isn't particularly good with strings.

```
sp {top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*two
  (state <s> ^name top-state
     ^operator <o>
     ^lexicon <lexicon>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
  (<i> ^token.lex << feel feeling think >>
       ^token.lex << about of >>
       ^token.lex <word1>
       ^token.lex {<word2> <> <word1>})
  (<lexicon> ^{<domain1> <> states <> emotion-responsibility-states}.<word-int1> <word1>
             ^{<domain2> <> states <> emotion-responsibility-states}.<word-int2> <word2>
             ^states.<state> <word-int1>
             ^states.<state> <word-int2>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem> + & )
  (<sem> ^type question
         ^q-slot value
         ^priority 2
         ^prop <sem1>)
  (<sem1> ^attribute feeling-about-state
          ^cause <state>
          ^type state)}
```

Figure 6.1: Soar rule which maps the utterance "How do you feel about [state]?" to the proper semantics

The original lexicon is divided into different types of entries, like `attributes`, `values` and `actions`. This is extended with `states`, `emotions`, `emotion-responsiblity-states` and `misc`. Part of the extension of the lexicon can be seen in figure 6.2. Of particular interest is the `states` entry. Every state is followed by all the words which define the state. Unfortunately, not all the naming conventions are consistent within MRE, so some percularities do arise. The phrase `minor-injuries`, for instance, is used both as a value and a state. The original lexicon maps several words to the *value* `minor-injuries`, which in our code is then used to detect the *state* `minor-injuries`. Furthermore, the same word can lead to different synonyms (for instance the word "secure" can

lead to the attribute `safety`, the value `secure` and the action `secure`). This can lead to variation in state detection, as no further criteria for selecting a particular synonym is used. Excluding the `actions` entry in the search is a first step in resolving such ambiguities, but it is noted that additional mechanisms might be needed when extending the lexicon.

```
sp {top-state*elaborate*state*add-lexicon-entries*emotions
  (state <s> ^name top-state
     ^agent-name << sgt medic director >>
     ^lexicon <lexicon>)
  -->
  (<lexicon> ^states <states>
     ^emotions <emotions>
     ^emotion-responsibility-states <emotion-responsibility-states>
              ^misc <misc>)
  (<states> ^maintain-goodwill crowd + &, goodwill + &
                    ^crowd-angry crowd + &, |Anger| + &
                ^boy-dead boy + &, dead + &
                    ^boy-healthy boy + &, healthy + &
                    ^minor-injuries minor-injuries + &, injuries + &, boy + &
                    ^serious-injuries serious-injuries + &, injuries + &, boy + &
                    ^critical-injuries critical-injuries + &, injuries + &, boy + &
                    ^driver-healthy driver + &, healthy + &
                    ^driver-minor-inj driver + &, minor-injuries + &, injuries + &
                    ^mother-healthy mom + &, healthy + &
                    ^know-boy-health boy + &, healthy + &, known + &
                    ^sqds-in-transit squads + &, transit + &
                    ^lt-in-transit lt + &, transit + &
                    ^lt-at-aa lt  + &, aa + &
                    ^lt-at-celic lt + &, celic + &
                     ....    )}
```

Figure 6.2: Soar rule which extends the lexicon with states

An issue already briefly pointed out in subsection 5.5.1 is the fact that the emotion code is rather restrictive on the understanding side. All current semantics take the emotion state attribute, like `max-feeling`, as the attribute value. This will in most cases result in only one matching emotion state (reference) for answering a question later on in the generation side, as described in subsection 4.4.2. In other words, selecting the state to use for an appropiate answer to a certain question is now done within the language understanding side, as opposed to the generation side. Future work could be focused on a more sophisticated answering mechanism, closer to the ones now used within MRE. One could think of losing the `max-feeling` state and when receiving the question "How do you feel?" creating semantics like `^attribute feeling ^object-id <self> ^q-slot value`. This will result in the generation of a set of references to all the `feeling` states, after which a complicated answer like "I'm feeling both worried and hopeful about the boy" is possible. One could also use intermediate attributes. An example of how this is done, is the use of the attribute `health-status`, which is mapped to all possible states concerning health, like `boy-dead` or `driver-minor-inj`. Due to lack of time, it was

not possible to make fully use of referencing. The current implementation does contain the foundations for further work, though. Note that state referencing can also be used for making references within a conversation, like asking "How do you feel about him?".

An example of a keyword mapping Soar rule can be found in figure 6.1. The rule searches for a couple of words used in asking a question like "How do you feel about [state]?" and two words which could lead to a state. These two words are mapped to the internal representation `word-int`, so for instance "kid" becomes `boy`. The two standardized words together lead to a certain state, for instance `boy` and `healthy` lead to the state `boy-healthy`. Initially, all entries were searched, which lead to problems during testing. Therefore, next to the already mentioned `actions` entry, the `state` and `emotion-responsibility-states` entries are excluded from the search, as these are ment for a final rather than intermediate result. All possibilities are added as an `emotion-semantics` attribute to the `interpretation` attribute. Certain Soar rules will select one `emotion-semantics` as the `semantics` that will continue the rest of the flow within the system.

6.4 Natural language generation

For the natural language generation part both Soar and TCL are used. The Soar rules detect the agent's intention to utter something and the particular rule that fires will gather all the necessary information needed to generate an output. This information is given to a TCL script which returns the utterance.

The left-hand side of a Soar rule first tries to detect a certain intention to say something. When implementation started the `comgoal` within the `output-speech` operator was used, but it turned out that it's structure would change, according to whether the agent would want to make an assertion or an answer. Consequently, the `goal` attribute is used instead. Recall from subsection 4.4.2 that the goal can be seen as the "why" and the communicative goal as that "what" in answering a question. Early during the implementation phase the decision of which emotion state to use was made by looking at the semantics of the original question, stored in `goal`. Later, it was learned that existing mechanisms within the MRE system automatically generate reference states for every state that can be used to answer the question (with the exception of questions for which on the fly states are used). These references are now used, but as explained in section 6.3 the mechanism is not used to it's full potentional.

Depending on the type of question and answer, certain information is needed in order to construct that answer. There are several places from where that information is gathered. All natural language is found in the look-up table `emia-template-filling`. The different strings of language are stored in attributes per category, as described in section 5.3.2. For the intensity, both the lower and the upper bound are stored, making it possible for other code to check on both. The agent-dependent template fillings for the responsible agent are added in seperate rules, because the agent's name has to be checked here. For the selection of the right words, information from the emotion and regular states is used. Personality information is taken from `io.input-link`. In the case a cause or responsible agent could be either present or absent, all possibilities are checked and the one with the highest information value receives the highest

priority.

The right-hand side produces a `surface` attribute which holds the answer as a value. An existing process will make sure this `surface` value will be spoken by the agent. In order to differentiate between certain surfaces, a priority system is used in some cases. This makes it possible to select a surface, before giving it a certain Soar preference. This priority system was introduced, because the why-questions originally interfered with the existing MRE code.

The value for the `surface` is generated by a template in TCL. All the necessary information is provided by Soar to TCL as parameters. Based on these parameters it selects a certain template alternative, as discussed in section 5.3.1. During implementation it was decided that the answer would be deterministic for the sake of time.

Chapter 7

Testing

7.1 Introduction

This chapter describes the testing phase of the assignment. The combined unit and integrations tests can be found in section 7.2. User tests will be discussed in section 7.3. These sections will only discuss how the tests were being performed and what the results are. All the test plans can be found in appendix C.

7.2 Unit and integration tests

7.2.1 Method

Unit testing is about testing the smallest unit of the software, in this case the Soar rule. Integration testing tests if the implemented software works correctly together. These tests are combined, because the smallest testable unit is the Soar rule, but certain rules depend on other rules.

The tests executed are white-box, in a sense that every Soar rule is added individually and is checked for it's results by looking at the created WME's in the Soar system. For integration testing the use-based testing method is used: in the first phase the Soar rules that do not depend on other rules (*independent rules*) but the original code are tested. The rules tested in the second phase only depend on rules in the first phase, rules tested in the third phase depend on rules in the first two phases, and so forth. The rules in Phase 5 produce the final output and will therefore be tested as a black-box, i.e. we look at the actual outcome, rather than any internal values. The testing environment is the initialized agent for the sergeant character (vital information, for some code is agent-dependant). The original MRE code is assumed correct.

7.2.2 Results

The tests resulted in a couple of issues. For minor ones like spelling errors within templates we point to the appendix. Others will be discussed here.

In general, it showed that the emotion code is often in battle with the existing code, as the NLU cannot dynamically be shut down. A rule was added which states that if there are more speech acts, the one generated by the emotion code

is preferred. Furthermore, our "Why do you feel [emotion]?" question interfered with the generic why-question, causing unexpected answers. This was due to the fact that for this particular question three Soar rules existed, all with a different preference. When the rule with the lowest preference was selected as a candidate by the system, it was always competing with existing rules which also had the lowest preference. In order to avoid these situations, priorities for surfaces were introduced which will first select the appropiate emotion rule, after which the selected rule receives a preference high enough not to interfere with existing rules.

The use of the existing follow-up why-question led to some problems too. Most of the time, a proper answer wouldn't come up and if it did, it was always constructed in the present tense, even if the state was to happen in the future. The first issue was due the way the code behind the why-question was getting the proper information and, again, the fact that two process race each other. The system would copy information neede for answering the why-question as soon as it was ready, not noticing that that information could be rewritten by another process later on. This will be taken care of by the MRE team, and should also fix a small bug where the agent sometimes utters "nothing" next to the answer about a world state. The tense issue required some work in the NLG module by the MRE team, and an additional Soar in the emotion code setting a time attribute to future where necessary. It is noted that the follow-up why-question cannot be used in combination with the on-the-fly states.

A problem with the `feeling` and `max-feeling` emotion states arose when more sources with equal intensity were available. To repair the fact that the `source` object associated with the causality (`cause`) was not properly created, an extra Soar rule has been added. The rule creates *only* the `source` attribute for an emotion state, picking one randomly if more options are available. The original rule has been altered so that if a `source` attribute exists, this object is used to add the `cause` attribute.

State detection didn't always go according to plan. This is due to how the original lexicon was set up, namely diveded into various entries. As several entries must be used by the emotion code in order to find the necessary synonyms, the code scans all the entries in the lexicon. By adding a specific `states` entry, this too is included in the search and so "boy" can lead to "minor-injuries" and from there either to the state `minor-injuries` or `driver-minor-inj`. To avoid these situations these rules has been changed, so it will exclude the `states` entry. Also, the `emotion-responsibility-states` and `actions` entries are excluded as these led to conflicts and ambiguities when using emotions for state detection.

Two bugs remain unsolved. The first one is that when emotion states are being updated during answering an emotion question, the agent won't be able to answer properly. This is due to the design of how the emotion states are created in combination with the use of references. When a question is received, references to states are produced, which will be used later on in the generation side. In the time between creating the references and using them, the states can be recreated, though, in order to match a new situation. This results in invalid references. In order to adress this problem, the emotion states should be created only once (O-support) and be updated by additional rules (I-support). A second problem arises as a result of a bug within the original MRE code. It came to light that the Terseness attribute isn't always present, due to an

impasse during initialization. As this attribute is needed for determining the template alternative, it's absence will cause the agent to fail generating a proper answer. The impasse is rare, though, and is beyond our abilities to fix.

7.3 User tests

For now, the emotion dialogue isn't meant for real use by army cadets, but rather as a feature for demonstrating the underlying emotional model of the agents. We therefore handled the user test quite informally. A user was given the possibility to 'play' with the system. We would observe the user, on what questions he asked, which phrasing he used and whether he found the answers sufficient and natural. Afterwards, our observations were discussed with the user. A total of three users participated in a user test.

7.3.1 User 1

During the first user test a couple of things came to light. Firt of all, the user asked a couple of questions which weren't covered by our code. These can be split up in questions which we do cover but use different phrasing for:

- How is it going?
- What's on your mind?
- How are you feeling?
- Who are you mad at?
- What are you worried about?

and sentences which semantics aren't covered:

- You don't look calm.
- Why are you mad at the mother?
- Why else? (follow up question)
- Are you feeling OK?
- Are you feeling bad?

We decided to implement the first type, as this would enhance the usability and required little effort. The second type of question would either require new states or advanced use of references. For instance, the last two questions would require designing a state dividing all emotions in good and bad ones. The second question requires states concerning emotions towards other agents. As these type of questions weren't covered in the initial analysis and design and would take quite some effort to implement and test, we decided not to implement these type of questions.

After this user test we mapped the word "sad" to Distress and "mad" to anger.

7.3.2 User 2

Most issues in the first user test were addressed for our second user test. Again the fact that references to earlier conversation parts are not possible was remarked. Furthermore, it became apparent that users can ask question negation wise, as in "Why are you not afraid?" We will save this type of questions for further research.

7.3.3 User 3

Again, the lack of context was noted. The user, not familiar with the MRE system, furthermore expected the emotions to be associated directly to certain personas rather than to states associated with personas. This, of course, is something dictated by the original MRE system, but could be an interesting point of discussion. As in the first user test, some types of questions were asked which are not covered, like "Why does the medic make you hopeful?", "Are you anxious about the boy?" and "Does the medic bring joy?". Again, we leave these for further research. Finally, the synonym "scared" was used and has been added to the lexicon.

Chapter 8

Conclusion

Recall from chapter 3 that the goal of this assignment was to make a scenario-independent implementation of an emotion dialogue, where agents can:

- understand user's utterances concerning emotions;
- use existing knowledge of their emotional state to express themselves; and
- express themselves using natural language.

These requirements were met: a user can ask a range of emotion related questions and receive answers to these questions, all in natural language. During the implementation, various limitations were encountered, creating opportunities for further research and fine-tuning – these recommendation can be found in chapter 9. This chapter concludes the assignment by looking at the different segments of the final implementation.

Knowledge about the emotions is made available by creating *emotion states*. The information needed for the analyzed questions are filtered out of the existing appraisals and structured into states with a conventional structure; the creation is scenario-independent. The translations of this information into strings of natural language is stored in *look-up tables*.

By preparing the necessary information in this way, some information is lost. For instance, the emotion state for a feeling, say `Distress`, only has the state it is most distressed about as a cause; all states an agent feels less distressed about are ignored. This can lead to odd conversations like:

"How do you feel about the driver?"
–"I'm worried about him, sir."
"Why are you worried?"
–"I'm worried, because the boy has critical injuries."

Because of limitations of the original natural language understanding module, a new design involving *keywords* has been implemented. The number of keywords per question was kept at a minimum to be as little restrictive as possible. After scanning, the most complete mapping is translated into semantics compliant to the system's conventions. In this way, the emotion code still utilizes the available language reasoning model.

A limitation using keywords is the fact that subtleties are hard to detect. For instance, negation-wise questions like "Why are you not happy?" would require to duplicate all existing rules. In addition, in order to be deterministic in detecting the correct questions, the number of keyword increases rapidly when increasing the number of questions.

For the language generation a system using *templates* has been designed, since the original NLG was too domain-specific to use. The choice of templates depends on various variables supplied by the agent. The templates are filled with the strings of natural language in the look-up tables mentioned earlier.

Unfortunately, a system using templates is very domain-specific, too. In order to provide the agent with the ability to utter various types of utterances in a natural way, a lot of templates are needed. We've chosen to give priority to natural and correct language for most cases and small discrepancies for others over correct but innatural language for all cases.

In order to use keyword scanning and templates in other scenario's a little effort is required. On the understanding side the new states must be made known, i.e. for every state certain keywords that detect that state must be given. For generating language concerning these new states, verbal references must be made known, i.e. for every state some strings of natural language must be given to use when referring to a certain state in an utterance.

During the assignment, some interesting points concerning emotion dialogue came to attention. A discussion was started about how to blend emotion dialogue naturally into existing dialogue. This discussion can be split into two main topics:

- How do humans in general use emotions in conversations?
- If and in what way is emotion dialogue used within an army environment?

The first question leads to aspects like the relation one has with its conversational partner, one's personality and one's coping strategies used. One could think of using emotions as a threshold to bring up certain conversational topics or using coping strategies for selecting an answer; denial, for instance, might then trigger a factual false, but emotional true answer. As for the use of emotion dialogue in the army, one could think of debriefing situations, or media training.

A major part in integrating emotion utterances in the dialogue is to make use of the dialogue context, like asking "How do you feel about that?". The reference system should be fully implemented for this.

A point of interest too, can be appraising emotion states. That would result in the agent being aware of his own emotions and react accordingly. Currently, the emotion code does alter the emotion intensity somewhat, but not much attention has been given to this aspect.

All in all, a lot of work remains to be done. Still, this assignment was worthwhile. The current result can be used for demonstrating the agent's emotions in a natural way and can, with a little effort, be extended to other scenarios. Also, it resulted in valuable insights for further work, of which an overview can be seen in the next chapter.

Chapter 9

Recommendations

A couple of aspects can be pointed out for future work:

Make full use of references. References are used to gather all states which could be used for answering a question. A decision mechanism then uses these states to come up with an aswer by, for instance, using one or more states information sources, or by deciding more information is needed in order to answer the question correctly. References can be used to create context ("How do you feel about him?" will come up with all the possible references for "him") and to create more sophisticated answers ("How do you feel?" will come up with all the matching feelings after which one or combination of these can be selected). In the original MRE system references are used for a couple of different types, like states and events. Using it for emotions would require designing and implementing a new type. Also, it would require to redesign and extend the number of emotion states in order to have a larger set for references.

Use coping strategies for generating agent's utterances. The emotion code currently always delivers an answer which is true. Of course, real people are often lying bastards. This can be realized for agents by using coping strategies for generating answers. For instance, when an agent uses a denial strategy when angry and is being confronted with "You seem angry..." it could reply with the opposite of the truth like "I am NOT angry!". For other emotions it could trigger empathatic reactions or problem solving reactions.

Use focus mechanisms for generating agent's utterances. In order to generate emotional verbal expressions the existing focus mechanism could be used. When for instance the dialogue involves the boy, this could trigger the agent in saying he is worried about the boy or suggesting appropiate action.

Extend known utterances. The known utterances could be extended with a couple of new ones, like "Why are you mad at the mother?". This would require new natural language understanding and generation code, and new states. For this particular question, one could think of designing new states, which for every person holds reasons why a particular emotion

is directed towards that individual. One could also think of extending the semantics, so it would be possible to dig deeper in the already known information. Then, instead of looking at the cause attribute for finding the cause of an emotion, one could check if that cause involves a certain individual. Other possible utterances are "Are you okay?", "How angry are you?", "Do you like the medic?" and "Why are you not happy?"

Use NLU and NLG module. Keyword scanning and templates can be used when aiming for a small and known set of utterances. When increasing the utterances, more and more effort goes into proper keyword detection and templating. It is therefore recommended to switch to a more generic approach, like the already existing NLU and NLG module. If that is not possible, it is recommended to use more parameters for template selection, like personality or coping traits, gender of subject and verb type. This will make an agent's utterances more dynamic and natural. Special attention could be given to include army hierarchy within a conversation.

Use emotion dialogue and states to influence emotions. Emotions are influenced by a lot of parameters, including emotions itself. In terms of the MRE system, emotions should be able to trigger an emotional change both internally and externally. Changes could be the results from the internal process of appraisal, requiring rules how to appraise a certain emotional state. As people are not always aware of their emotions, agents too should not be appraising their own emotional state all the time. Rather, certain conditions could lead to this, like reaching a critical intensity of a certain emotion. Secondly, talking about emotions could be a trigger to change one's emotion, as emotional dialogue forces people to examine their emotions, thus making them explicit. Both methods would result in either intensifying or soothing a certain emotion.

Fix known bugs. Two bugs remain unsolved. The first one is that when emotions states are being updated during answering an emotion question, the agent won't be able to answer properly. This is due to the design of how the emotion states are created in combination with the use of references. When a question is received, references to states are produced, which will be used later on in the generation side. In the time between creating the references and using them, the states can be recreated, though, in order to match a new situation. This results in invalid references. In order to adress this problem, the emotion states should be created only once (O-support) and be updated by additional rules (I-support). A second problem arises as a result of a bug within the original MRE code. It came to light that in rare cases the Terseness attribute isn't present, due to an impasse during initialization. As this attribute is needed for determining the template alternative, it's absence will cause the agent to fail generating a proper answer.

Appendix A

Speech acts

This appendix summarizes the type of speech acts which are currently covered by the emotion code. As question recognition relies on detecting certain keywords, only part of all possible questions will be mentioned. The answers too form only a subset of the possible answers; the actual answers depend on the state of the agent. Details can be found in the Soar and TCL code in appendix B.

It is noted that when engaging in emotion dialogue, the dialogue will be most interesting when the terseness value is high. The agent will then answer in a short manner, making follow up questions possible. Also, it is noted that when typing questions, one should restrict to lower case characters only.

The last section of this appendix shows the synonyms used for the emotions.

A.1 Emotion type

A.1.1 Class 1

Questions:

- How are you?
- How is it going?
- How do you feel?
- What's wrong?

Answers:

- I'm feeling pretty hopeful.
- I'm feeling pretty hopeful, because the boy will probably be treated at the hospital.
- I'm fine, sir.

A.1.2 Class 2

Questions:

- Do you feel angry?
- Are you worried?
- You seem distressed.

Answers:

- Not at all, sir.
- Yes, sir.
- Quite a bit, sir.
- Yes, sir, because the boy has critical injuries.

A.1.3 Class 3

Questions:

- How do you feel about the boy
- What do you think of the boy being injured?
- How are you feeling about our team splitting up?

Answers:

- I'm feeling worried about him, sir.
- I'm feeling angry about it, sir.

A.2 Intensity

Questions:

- Calm down, you maniac!
- Relax, dude.

Answers:

- Will do, sir.
- I am calm, sir.

A.3 Emotion state

Questions:

- Why are you worried?
- Why do you feel hopeful?
- What's causing you to be sad?

Answers:

- I'm not feeling angry, sir.
- That's personal, sir.
- Our team will probably split up, that's your fault, sir.
- The boy has critical injuries, sir.

A.4 Emotion responsibility

Questions:

- Who's responsible for making you feel angry?
- Who are you mad at?
- Who makes you feel sad?
- Towards whom do you feel guilty?

Answers:

- I'm not feeling angry, sir.
- That is Tucci's fault, sir.
- Lopez, sir
- No one in particular, sir.

A.5 Emotion synonyms

Natural language input	**Natural language output**
anger, angry, annoyed, pissed, mad	angry
anxiety, anxious	anxious
distress, distressed, upset, disturbed, troubled worried, concerned, sad	worried
fear, fearful, afraid, frightened, stressed pessimistic, scared	afraid
guilt, guilty	guilty
hope, hopeful, optimistic	hopeful
joy, joyful, exited, happy	joyful

Appendix B

Soar and TCL code

This sections contains the Soar and TCL code for this assignment. Every file has “emia” (EMotion InterAction) in it’s name as a recognition tag. These files can be found on http://mullert.adsl.utwente.nl/ stage/soarcode.

B.1 Emotion-states-emia.soar

```
### Copyright 2003 Arno Hartholt  |Tijmen Joppe Muller
### Institute for Creative Technologies
###############################################################################
###
### File              : emotion-states-emia.soar
### Original author(s): Tijmen Joppe Muller (tijmen@avpec1910.nl)
###                     Arno Hartholt (d.o.a.hartholt@student.utwente.nl)
### Supervisor        : Jonathan Gratch (gratch@ict.usc.edu)
### Organization      : Institute for Creative Technologies
### Created on        : November 13, 2003
### Last Modified By  : Arno Hartholt
### Soar Version      : 7
### Documentation     : Interaction on emotions
###                     (http://mullert.adsl.utwente.nl/~stage/reports/emia.pdf)
###_____________________________________________________________________________
### HISTORY
###
### 11-13-03 [TJM]   Document created, initialization section added.
### 11-17-03 [TJM]   Rules top-ps*emia*elaborate*emotion-state*max-feeling and
###                  -*feeling added.
### 11-18-03 [TJM]   Rules top-ps*emia*emotion-state*add*causality and
###                  -*to*negated added.
### 11-19-03 [TJM]   Rules top-ps*emia*elaborate*emotion-state*responsible-for-
###                  feeling added
### 11-25-03 [TJM]   Rule top-ps*emia*emotion-state*feeling-about-state*add*
###                  sim-object
### 12-04-03 [TJM]   Rule top-ps*emia*emotion-state*add*fake*causality
###                  Changes in the lookup-table (influence-status)
### 12-16-03 [AH]    Initialization section moved to language-emia.soar
### 12-19-03 [AH]    Changed rules top-ps*emia*emotion-state*add*cause and
###                  top-state*emia*emotion-state*feeling-about-state so cause
###                  is set tothe actual objects instead of only the names
###_____________________________________________________________________________
### USE
###
### Implements natural language understanding and generation, using emotion states
```

```
### and templates.
###_____________________________________________________________________________
### ISSUES
###
### At this moment, the causality relationship has only a single value, which
### seems not right, since a feeling (i.e. anger) can have more than one reason.
### As a result, the simulation object also has a single value.
###
### For the cause attribute the state that has the most effect on a certain
### emotion is taken as the value. This should be done in a better way, i.e.
### if state A had 0.50 effect on the anger emotion and state B has 0.49 effect,
### it doesn't seem right to neglect state B completely. Possibly calculation
### of probability should be used.
###_____________________________________________________________________________
### KNOWN BUGS
###
###_____________________________________________________________________________
### LIMITATIONS
###
###_____________________________________________________________________________
### EXTERNAL REFERENCES
###
### Used in combination with language-emia.soar and lexicon-emia.soar
###_____________________________________________________________________________
### TO DO
###
###_____________________________________________________________________________
### SUMMARY
###
###
###############################################################################

echo "\nLoading emotion-states-emia.soar\n"

###############################################################################
# Emotion states
# ~~~~~~~~~~~~~~

# Create the emotion states for the most important feeling. The second emotion
# state is just the negation of the first.

sp {top-ps*emia*elaborate*emotion-state*max-feeling
    (state <s> ^agent-name <name>
       ^current-state <cs>
       ^name top-state
       ^io.input-link.self.max <maxfeeling>)
    -->
    (<cs> ^<new-emotion-state1> <nes1>
  ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute max-feeling
    ^belief true
    ^id <new-emotion-state1>
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity positive
    ^satisfied true
    ^type state
    ^value <maxfeeling>)
    (<nes2> ^attribute max-feeling
    ^belief false
```

```
    ^id <new-emotion-state2>
    ^initial-belief false
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity negative
    ^satisfied false
    ^type state
    ^value <maxfeeling>)}


# Create a feeling state for every emotion. If the intensity of the emotion
# is greater or equal to 0.2, the agent beliefs he feels that emotion (which
# means the polarity attribute has value 'positive'), if
# smaller than 0.2 he does not.

sp {top-ps*emia*elaborate*emotion-state*feeling*true
    (state <s> ^agent-name <name>
       ^appraisals.types.type <emotion>
       ^current-state <cs>
       ^name top-state
       ^io.input-link.self <self>)
    (<self> ^<emotion> {<intensity> >= 0.2})
    -->
    (<cs> ^<new-emotion-state1> <nes1>
  ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute feeling
    ^belief true
    ^id <new-emotion-state1>
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity positive
    ^satisfied true
    ^type state
    ^value <emotion>)
    (<nes2> ^attribute feeling
    ^belief false
    ^id <new-emotion-state2>
    ^initial-belief false
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity negative
    ^satisfied false
    ^type state
    ^value <emotion>)}

sp {top-ps*emia*elaborate*emotion-state*feeling*false
    (state <s> ^agent-name <name>
       ^appraisals.types.type <emotion>
       ^current-state <cs>
       ^name top-state
       ^io.input-link.self <self>)
    (<self> ^<emotion> {<intensity> < 0.2})
    -->
    (<cs> ^<new-emotion-state1> <nes1>
  ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute feeling
    ^belief true
    ^id <new-emotion-state1>
```

```
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity negative
    ^satisfied true
    ^type state
    ^value <emotion>)
    (<nes2> ^attribute feeling
    ^belief false
    ^id <new-emotion-state2>
    ^initial-belief false
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity positive
    ^satisfied false
    ^type state
    ^value <emotion>)}


# Add the 'cause' attribute to the 'feeling' and 'max-feeling' emotion states.
# This is done only if the agent actually feels the emotion, i.e. polarity
# is positive and belief is true.
# The rule checks for the appraisal that has the biggest influence on a
# emotion and places a link to the appropriate state in the emotion state.
# The appraisal that is the source is saved in the attribute with the same
# name. If two appraisal frames have the exact same influence, one of them is
# picked arbitrarily. If no appraisal state can be found with a certain
# emotion, no causality is added.
#
# <emotion>                 a certain emotion
# <emotion-appraisal-max>   appraisal object with the largest intensity on
#                           the emotion defined by <emotion>, is saved in the
#                           ^source attribute
# <es>                      emotion state 'feeling' or 'max-feeling' which is
#                           about emotion defined by <emotion>
# <state>                   the state that is appraised by
#                           <emotionappraisalmax>
# <emotion-appraisal>       any appraisal on emotion defined by <emotion>

sp {top-ps*emia*emotion-state*add*source
    (state <s> ^agent-name <name>
       ^appraisals <appr>
       ^current-state <cs>
       ^name top-state)
    (<appr> ^types.type <emotion>
    ^<emotion> <emotion-appraisal-max>)
    (<cs> ^<emotion-state-obj> <emotion-state>)
    (<emotion-appraisal-max> ^agent <name>
     ^appraisal.object.name <state>
     ^context self
     ^intensity <em-intensity-max>)
    (<emotion-state> ^attribute << feeling max-feeling >>
     ^belief true
     ^polarity positive
     ^value <emotion>)
  -{(<appr> ^<emotion> <emotion-appraisal>)
    (<emotion-appraisal> ^agent <name>
 ^context self
 ^intensity {<em-intensity> > <em-intensity-max>})}
    -->
```

```
    (<emotion-state> ^source <emotion-appraisal-max> + =)}


sp {top-ps*emia*emotion-state*add*cause
    (state <s> ^current-state <cs>
       ^name top-state)
    (<cs> ^<emotion-state-obj> <emotion-state>)
    (<emotion-appraisal-max> ^appraisal.object.state-obj <state>)
    (<emotion-state> ^attribute << feeling max-feeling >>
     ^belief true
     ^polarity positive
     ^source <emotion-appraisal-max>)
    -->
    (<emotion-state> ^cause <state>)}


# This rule makes sure that the negated state of 'feeling' and 'max-feeling'
# have the same cause and source attribute.

sp {top-ps*emia*emotion-state*add*causality*to*negation
    (state <s> ^agent-name <name>
       ^current-state <cs>
       ^name top-state)
    (<cs> ^<emotion-state-obj> <emotion-state>)
    (<emotion-state> ^attribute << feeling max-feeling >>
     ^cause <cause>
     ^negation <negated-emotion-state>
     ^source <source>)
    -->
    (<negated-emotion-state> ^cause <cause>
     ^source <source>)}


# Create a 'responsible-for-[emotion]' state for every emotion. The creation of
# this state depends on the 'feeling' states, since it is only created if the
# agent actually feels the emotion, e.g. intensity is greater than the
# treshold.

sp {top-ps*emia*elaborate*emotion-state*responsible-for-anger
    (state <s> ^agent-name <name>
       ^current-state <cs>
       ^name top-state)
    (<cs> ^<emotion-state-obj> <emotion-state>)
    (<emotion-state> ^attribute feeling
     ^belief true
     ^source <appraisal-obj>
     ^value |Anger|)
    (<appraisal-obj> ^appraisal.evaluation.responsible-agent <resp-agent>
     ^appraisal.evaluation.evaluation <blame>)
    -->
    (<cs> ^<new-emotion-state1> <nes1>
  ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute responsible-for-Anger
    ^belief true
    ^cause <blame>
    ^id <new-emotion-state1>
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity positive
    ^satisfied true
```

```
    ^type state
    ^value <resp-agent>)
    (<nes2> ^attribute responsible-for-Anger
    ^belief false
    ^cause <blame>
    ^id <new-emotion-state2>
    ^initial-belief false
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity negative
    ^satisfied false
    ^type state
    ^value <resp-agent>)}

sp {top-ps*emia*elaborate*emotion-state*responsible-for-anxiety
    (state <s> ^agent-name <name>
       ^current-state <cs>
       ^name top-state)
    (<cs> ^<emotion-state-obj> <emotion-state>)
    (<emotion-state> ^attribute feeling
     ^belief true
     ^source <appraisal-obj>
     ^value |Anxiety|)
    (<appraisal-obj> ^appraisal.evaluation.responsible-agent <resp-agent>
     ^appraisal.evaluation.evaluation <blame>)
    -->
    (<cs> ^<new-emotion-state1> <nes1>
  ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute responsible-for-Anxiety
    ^belief true
    ^cause <blame>
    ^id <new-emotion-state1>
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity positive
    ^satisfied true
    ^type state
    ^value <resp-agent>)
    (<nes2> ^attribute responsible-for-Anxiety
    ^belief false
    ^cause <blame>
    ^id <new-emotion-state2>
    ^initial-belief false
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity negative
    ^satisfied false
    ^type state
    ^value <resp-agent>)}

sp {top-ps*emia*elaborate*emotion-state*responsible-for-distress
    (state <s> ^agent-name <name>
       ^current-state <cs>
       ^name top-state)
    (<cs> ^<emotion-state-obj> <emotion-state>)
    (<emotion-state> ^attribute feeling
     ^belief true
     ^source <appraisal-obj>
```

```
     ^value |Distress|)
    (<appraisal-obj> ^appraisal.evaluation.responsible-agent <resp-agent>
     ^appraisal.evaluation.evaluation <blame>)
    -->
    (<cs> ^<new-emotion-state1> <nes1>
  ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute responsible-for-Distress
    ^belief true
    ^cause <blame>
    ^id <new-emotion-state1>
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity positive
    ^satisfied true
    ^type state
    ^value <resp-agent>)
    (<nes2> ^attribute responsible-for-Distress
    ^belief false
    ^cause <blame>
    ^id <new-emotion-state2>
    ^initial-belief false
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity negative
    ^satisfied false
    ^type state
    ^value <resp-agent>)}

sp {top-ps*emia*elaborate*emotion-state*responsible-for-fear
    (state <s> ^agent-name <name>
       ^current-state <cs>
       ^name top-state)
    (<cs> ^<emotion-state-obj> <emotion-state>)
    (<emotion-state> ^attribute feeling
     ^belief true
     ^source <appraisal-obj>
     ^value |Fear|)
    (<appraisal-obj> ^appraisal.evaluation.responsible-agent <resp-agent>
     ^appraisal.evaluation.evaluation <blame>)
    -->
    (<cs> ^<new-emotion-state1> <nes1>
  ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute responsible-for-Fear
    ^belief true
    ^cause <blame>
    ^id <new-emotion-state1>
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity positive
    ^satisfied true
    ^type state
    ^value <resp-agent>)
    (<nes2> ^attribute responsible-for-Fear
    ^belief false
    ^cause <blame>
    ^id <new-emotion-state2>
    ^initial-belief false
```

```
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity negative
    ^satisfied false
    ^type state
    ^value <resp-agent>)}

sp {top-ps*emia*elaborate*emotion-state*responsible-for-guilt
    (state <s> ^agent-name <name>
       ^current-state <cs>
       ^name top-state)
    (<cs> ^<emotion-state-obj> <emotion-state>)
    (<emotion-state> ^attribute feeling
     ^belief true
     ^source <appraisal-obj>
     ^value |Guilt|)
    (<appraisal-obj> ^appraisal.evaluation.responsible-agent <resp-agent>
     ^appraisal.evaluation.evaluation <blame>)
    -->
    (<cs> ^<new-emotion-state1> <nes1>
  ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute responsible-for-Guilt
    ^belief true
    ^cause <blame>
    ^id <new-emotion-state1>
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity positive
    ^satisfied true
    ^type state
    ^value <resp-agent>)
    (<nes2> ^attribute responsible-for-Guilt
    ^belief false
    ^cause <blame>
    ^id <new-emotion-state2>
    ^initial-belief false
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity negative
    ^satisfied false
    ^type state
    ^value <resp-agent>)}

sp {top-ps*emia*elaborate*emotion-state*responsible-for-hope
    (state <s> ^agent-name <name>
       ^current-state <cs>
       ^name top-state)
    (<cs> ^<emotion-state-obj> <emotion-state>)
    (<emotion-state> ^attribute feeling
     ^belief true
     ^source <appraisal-obj>
     ^value |Hope|)
    (<appraisal-obj> ^appraisal.evaluation.responsible-agent <resp-agent>
     ^appraisal.evaluation.evaluation <blame>)
    -->
    (<cs> ^<new-emotion-state1> <nes1>
  ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute responsible-for-Hope
```

```
    ^belief true
    ^cause <blame>
    ^id <new-emotion-state1>
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity positive
    ^satisfied true
    ^type state
    ^value <resp-agent>)
    (<nes2> ^attribute responsible-for-Hope
    ^belief false
    ^cause <blame>
    ^id <new-emotion-state2>
    ^initial-belief false
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity negative
    ^satisfied false
    ^type state
    ^value <resp-agent>)}

sp {top-ps*emia*elaborate*emotion-state*responsible-for-joy
    (state <s> ^agent-name <name>
       ^current-state <cs>
       ^name top-state)
    (<cs> ^<emotion-state-obj> <emotion-state>)
    (<emotion-state> ^attribute feeling
     ^belief true
     ^source <appraisal-obj>
     ^value |Joy|)
    (<appraisal-obj> ^appraisal.evaluation.responsible-agent <resp-agent>
     ^appraisal.evaluation.evaluation <blame>)
    -->
    (<cs> ^<new-emotion-state1> <nes1>
  ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute responsible-for-Joy
    ^belief true
    ^cause <blame>
    ^id <new-emotion-state1>
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity positive
    ^satisfied true
    ^type state
    ^value <resp-agent>)
    (<nes2> ^attribute responsible-for-Joy
    ^belief false
    ^cause <blame>
    ^id <new-emotion-state2>
    ^initial-belief false
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity negative
    ^satisfied false
    ^type state
    ^value <resp-agent>)}
```

```
# Create the on-the-fly 'feeling-about-state' emotion state.
# The emotion state is created only if the question is under discussion, and
# therefor is an on-the-fly state. Also, it is only created if there actually
# is an appraisal about the state asked for. Finally, we apply a small treshold
# (not the same as for the aggregate emotion intensity!) to cut out
# insignificant appraisals.

sp {top-ps*emia*elaborate*emotion-state*feeling-about-state
    :i-support
    (state <s> ^agent-name <name>
       ^appraisals <appr>
       ^current-state <cs>
       ^name top-state
       ^operator <op>)
    (<appr> ^types.type <emotion>
    ^<emotion> <emotion-appraisal-max>)
    (<emotion-appraisal-max> ^agent <name>
     ^appraisal.object.state-obj <state>
     ^appraisal.object.name <name-obj>
     ^context self
     ^intensity {<em-intensity-max> > 5 })
  -{(<appr> ^<emotion> <emotion-appraisal>)
    (<emotion-appraisal> ^agent <name>
 ^appraisal.object.state-obj <state>
 ^context self
 ^intensity {<em-intensity> > <em-intensity-max>})}
    (<op> ^goal.content.content <c>
 ^name output-speech)
    (<c> ^type question
 ^q-slot value
 ^prop <prop>)
    (<prop> ^attribute feeling-about-state
    ^cause <name-obj>
    ^type state)
    -->
    (<cs> ^<new-emotion-state1> <nes1>
 ^<new-emotion-state2> <nes2>)
    (<nes1> ^attribute feeling-about-state
    ^belief true
    ^cause <state>
    ^id <new-emotion-state1>
    ^initial-belief true
    ^name <new-emotion-state1>
    ^negation <nes2>
    ^object-id <name>
    ^polarity positive
    ^satisfied true
    ^source <emotion-appraisal-max>
    ^type state
    ^value <emotion>)
    (<nes2> ^attribute feeling-about-state
    ^belief false
    ^cause <state>
    ^id <new-emotion-state2>
    ^initial-belief false
    ^name <new-emotion-state2>
    ^negation <nes1>
    ^object-id <name>
    ^polarity negative
    ^satisfied false
```

```
    ^source <emotion-appraisal-max>
    ^type state
    ^value <emotion>)}


# Add a simulation-object attribute to the feeling-about-state object, by
# copying it from the state the emotion state is about.

sp {top-ps*emia*emotion-state*feeling-about-state*add*sim-object
    (state <s> ^current-state <cs>
       ^name top-state)
    (<cs> ^<emotion-state-obj> <emotion-state>
  ^<state-obj> <state>)
    (<emotion-state> ^attribute feeling-about-state
     ^cause <state-obj>)
    (<state> ^sim-object <sim-obj>)
    -->
    (<emotion-state> ^sim-object <sim-obj>)}
```

B.2 Language-emia.soar

```
### Copyright 2003 Arno Hartholt | Tijmen Joppe Muller
### Institute for Creative Technologies
##############################################################################
###
### File              : language-emia.soar
### Original author(s): Arno Hartholt (d.o.a.hartholt@student.utwente.nl)
###                     Tijmen Joppe Muller (tijmen@avpec1910.nl)
### Supervisor        : Jonathan Gratch (gratch@ict.usc.edu)
### Organization      : Institute for Creative Technologies
### Created on        : November 13, 2003
### Last Modified By  : Arno - no middle name - Hartholt
### Soar Version      : 7
### Documentation     : Interaction on emotions
###                     (http://mullert.adsl.utwente.nl/~stage/reports/emia.pdf)
###____________________________________________________________________________
### HISTORY
###
### 11-13-03 [AH]   Document created, keyword mapping & assertion rules try-outs
### 11-17-03 [AH]   Added template try-outs
### 11-26-03 [AH]   Extended templates, and keyword mapping and assertions rules
### 12-04-03 [AH]   Introduced emotion-semantics and selection rules
### 12-15-03 [AH]   Added additional keywords
### 12-16-03 [AH]   Added [TJM]'s look-up table rules (initialization) from
###                 emotion-states-emia.soar
###                 Changed natural language for emotions
### 12-19-03 [AH]   Started using references
### 12-22-03 [AH]   Introduced priorities in speech-acts, so we can choose between
###                 rules and give the winner a certain pref.
### 12-23-03 [AH]   Introduced priorities in surfaces for why-questions
### 12-24-03 [AH]   Fixed some reference stuff
###                 Added selection based on emotion intensity for ambiguous
###                 'about' questions
### 01-05-03 [AH]   Added object-id to every semantics
### 01-12-03 [AH]   Updated state list in order to match the current configuration
###
###____________________________________________________________________________
### USE
###
### This file is used for emotion dialogue. It sets up a look-up table for
### natural language, it scans for keywords in user's questions and it produces
```

```
### answers to that questions.
###
###_____________________________________________________________________________
### ISSUES
###
### The Terseness attribute (io.input-link) sometimes isn't set due to an impasse
### during MRE initialization. This prevents the assertion rules from firing.
### Restart the agent and pray.
###
### As feeling-about-state states are generated on the fly, the system does not
### generate references when this type of question is asked. The assertion rules
### for these questions are therefore implemented without the use of references.
### This means, among others, that ambiguity is resolved in understand-speech,
### rather than in output-speech.
###_____________________________________________________________________________
### KNOWN BUGS
###
### Emotion questions don't work if they're asked at the same time the emotion
### intensity changes. In other words: when emotions change, the system needs
### some time before it can answer emotion questions. This is due to the fact that
### the emotion states are recreated when emotion intensity changes, making earlier
### references invalid. The states in emotion-states-emia.soar should be rewritten
### so that existing states get updated rather than recreated.
###_____________________________________________________________________________
### LIMITATIONS
###
###_____________________________________________________________________________
### EXTERNAL REFERENCES
###
### Used in combination with emotion-states-emia.soar and lexicon-emia.soar
###_____________________________________________________________________________
### TO DO
###
###_____________________________________________________________________________
### SUMMARY
###
###
################################################################################

echo "\nLoading language-emia.soar\n"

################################################################################
# INITIALIZATION
# ~~~~~~~~~~~~~~
#
# Produces a look-up table which the output-speech rules in this file use
# in order to get natural language.
#
################################################################################

# Set up the shell for the template fillings lookup table.
sp {top-ps*emia*elaborate*lookup-table*template
    (state <s> ^name top-state)
    -->
    (write (crlf) |emia.soar: Initialize lookup table template fillings | (crlf))
    (<s>   ^emia-template-filling <etf>)
    (<etf> ^nl-emotion <nle>
   ^nl-influence-status <nlis>
   ^nl-intensity-lower <nlil>
   ^nl-intensity-upper <nliu>
   ^nl-responsible_agent <nlr>
   ^nl-responsible_agent-poss <nlrp>
```

```
   ^nl-state-pre <nls1>
   ^nl-state-post <nls2>)}


# Add the static (agent-independent) template fillings to the lookup table.
sp {top-ps*emia*elaborate*lookup-table*static-content*template
    (state <s> ^name top-state
       ^emia-template-filling <etf>)
    (<etf> ^nl-emotion <nle>
   ^nl-influence-status <nlis>
   ^nl-intensity-lower <nlil>
   ^nl-intensity-upper <nliu>
   ^nl-responsible_agent <nlr>
   ^nl-responsible_agent-poss <nlrp>
   ^nl-state-pre <nls1>
   ^nl-state-post <nls2>)
    -->
    (write (crlf) |emia.soar: Elaborate lookup table, static content| (crlf))
    (<nle> ^|Anger|             |angry|
   ^|Anxiety|           |anxious|
   ^|Distress|          |worried|
   ^|Fear|              |afraid|
   ^|Guilt|             |guilty|
   ^|Hope|              |hopeful|
   ^|Joy|               |joyful|)
    (<nlil> ^0                  |"a little"|
    ^0.25               |pretty|
    ^0.5                |""|
    ^0.75               |very|
    ^1                  |very|)
    (<nliu> ^0.25               |"a little"|
    ^0.5                |pretty|
    ^0.75               |""|
    ^1.1                |very|)
    (<nlis> ^inhabitor          <nlis1>
    ^facilitator        <nlis2>)
    (<nlis1> ^confirmed         |"is not"|
     ^unconfirmed       |"probably won't be"|)
    (<nlis2> ^confirmed         |is|
     ^unconfirmed       |"will probably be"|)
    (<nlr> ^1sldr               |Johnson|
   ^2sldr               |"The second squad leader"|
   ^3sldr               |"The third squad leader"|
   ^4sldr               |Lopez|
   ^ambulance           |"The ambulance"|
   ^base                |"The base"|
   ^lt                  |"You are"|
   ^medevac             |"The MedEvac"|)
    (<nlrp> ^1sldr              |"Johnson's"|
    ^2sldr              |"second squad leader's"|
    ^3sldr              |"third squad leader's"|
    ^4sldr              |"Lopez's"|
    ^ambulance          |"the ambulance's"|
    ^base               |"the base's"|
    ^lt                 |your|
    ^medevac            |"the MedEvac's"|)
    (<nls1> ^maintain-goodwill  |"the goodwill of the crowd"|
    ^crowd-angry        |"the crowd"|
    ^boy-dead           |"the boy"|
    ^boy-healthy        |"the boy"|
    ^minor-injuries     |"the boy"|
    ^serious-injuries   |"the boy"|
```

```
^critical-injuries  |"the boy"|
^driver-healthy     |"the driver"|
^driver-minor-inj   |"the driver"|
^mother-healthy     |"the mother"|
^know-boy-health    |"the health of the boy"|
^sqds-in-transit    |"the squads"|
^lt-in-transit      |"the lieutenant"|
^lt-at-aa           |"the lieutenant"|
^lt-at-celic        |"the lieutenant"|
^sgt-at-aa          |"the sergeant"|
^medic-at-aa        |"the medic"|
^at-boy-aa          |"the boy"|
^mom-at-aa          |"the mother"|
^1st-sqd-at-aa      |"the first squad"|
^1st-sqd-at-celic   |"the first squad"|
^1st-sqd-at-lz      |"the first squad"|
^1st-sqd-in-transit |"the first squad"|
^2nd-sqd-at-aa      |"the second squad"|
^2nd-sqd-at-celic   |"the second squad"|
^2nd-sqd-at-lz      |"the second squad"|
^3rd-sqd-at-aa      |"the third squad"|
^3rd-sqd-at-celic   |"the third squad"|
^3rd-sqd-at-lz      |"the third squad"|
^4th-sqd-at-aa      |"the fourth squad"|
^4th-sqd-at-celic   |"the fourth squad"|
^4th-sqd-at-lz      |"the fourth squad"|
^4th-sqd-in-transit |"the fourth squad"|
^at-boy-hospital    |"the boy"|
^mom-in-intersect   |"the mother"|
^sqd-in-intersect   |"the squad"|
^medevac-at-aa      |"the MedEvac"|
^medevac-at-base    |"the MedEvac"|
^medevac-called     |"the MedEvac"|
^medevac-overhead   |"the MedEvac"|
^amb-at-aa          |"the ambulance"|
^amb-at-base        |"the ambulance"|
^ambulance-called   |"the ambulance"|
^secure-route       |"the route to Celic"|
^aa-secure          |"the assembly area"|
^accident-secure    |"the accident site"|
^12-to-4-secure     |"the assembly area"|
^4-to-8-secure      |"the assembly area"|
^8-to-12-secure     |"the assembly area"|
^lz-secure          |"the landing zone"|
^lz-marked          |"the landing zone"|
^lz-clear           |"the landing zone"|
^support-1-6        |"our team"|
^retain-mass        |"our team"|
^fracture-unit      |"our team"|
^1-6-at-celic |"our team"|
^hospital-at-tuzla  |"the hospital"|)
(<nls2> ^maintain-goodwill  |maintained|
^crowd-angry        |angry|
^boy-dead           |dead|
^boy-healthy        |healthy|
^minor-injuries     |"slightly injured"|
^serious-injuries   |"seriously injured"|
^critical-injuries  |"critically injured"|
^driver-healthy     |healthy|
^driver-minor-inj   |"slightly injured"|
^mother-healthy     |healthy|
^know-boy-health    |known|
```

```
    ^sqds-in-transit    |"in transit to the assembly area"|
    ^lt-in-transit      |"in transit to the assembly area"|
    ^lt-at-aa           |"at the assembly area"|
    ^lt-at-celic        |"at Celic"|
    ^sgt-at-aa          |"at the assembly area"|
    ^medic-at-aa        |"at the assembly area"|
    ^at-boy-aa          |"at the assembly area"|
    ^mom-at-aa          |"at the assembly area"|
    ^1st-sqd-at-aa      |"at the assembly area"|
    ^1st-sqd-at-celic   |"at Celic"|
    ^1st-sqd-at-lz      |"at the landing zone"|
    ^1st-sqd-in-transit |"in transit to the assembly area"|
    ^2nd-sqd-at-aa      |"at the assembly area"|
    ^2nd-sqd-at-celic   |"at Celic"|
    ^2nd-sqd-at-lz      |"at the landing zone"|
    ^3rd-sqd-at-aa      |"at the assembly area"|
    ^3rd-sqd-at-celic   |"at Celic"|
    ^3rd-sqd-at-lz      |"at the landing zone"|
    ^4th-sqd-at-aa      |"at the assembly area"|
    ^4th-sqd-at-celic   |"at Celic"|
    ^4th-sqd-at-lz      |"at the landing zone"|
    ^4th-sqd-in-transit |"in transit to the assembly area"|
    ^at-boy-hospital    |"treated at the hospital"|
    ^mom-in-intersect   |"at the intersection"|
    ^sqd-in-intersect   |"at the intersection"|
    ^medevac-at-aa      |"at the assembly area"|
    ^medevac-at-base    |"at the base"|
    ^medevac-called     |"called for"|
    ^medevac-overhead   |overhead|
    ^amb-at-aa          |"at the assembly area"|
    ^amb-at-base        |"at the base"|
    ^ambulance-called   |"called for"|
    ^secure-route       |secured|
    ^aa-secure          |secured|
    ^accident-secure    |secured|
    ^12-to-4-secure     |"partly secured"|
    ^4-to-8-secure      |"partly secured"|
    ^8-to-12-secure     |"partly secured"|
    ^lz-secure          |secured|
    ^lz-marked          |"marked by green smoke"|
    ^lz-clear           |"cleared of civilians"|
    ^support-1-6        |"active supporting Eagle 2-6"|
    ^retain-mass        |together|
    ^fracture-unit      |"split up"|
    ^1-6-at-celic |"at Celic"|
    ^hospital-at-tuzla  |"at Tuzla"|)}


# Add the dynamic (agent-dependent) template fillings to the loopup table, i.e.
# pronouns for the medic, mother, sergeant and self.

sp {top-ps*emia*elaborate*lookup-table*dynamic-content*medic
    (state <s> ^agent-name <name> <> medic
       ^emia-template-filling <etf>
       ^name top-state)
    (<etf> ^nl-responsible_agent <nlr>
   ^nl-responsible_agent-poss <nlrp>)
    -->
    (write (crlf) |emia.soar: Elaborate lookup table, dynamic content (medic)| (crlf))
    (<nlr> ^medic |Tucci|)
    (<nlrp> ^medic |"Tucci's"|)}
```

```
sp {top-ps*emia*elaborate*lookup-table*dynamic-content*mom
    (state <s> ^agent-name <name> <> mom
       ^emia-template-filling <etf>
       ^name top-state)
    (<etf> ^nl-responsible_agent <nlr>
   ^nl-responsible_agent-poss <nlrp>)
    -->
    (write (crlf) |emia.soar: Elaborate lookup table, dynamic content (mother)| (crlf))
    (<nlr> ^mom |"The mother"|)
    (<nlrp> ^mom |"the mother's"|)}

sp {top-ps*emia*elaborate*lookup-table*dynamic-content*sgt
    (state <s> ^agent-name <name> <> sgt
       ^emia-template-filling <etf>
       ^name top-state)
    (<etf> ^nl-responsible_agent <nlr>
   ^nl-responsible_agent-poss <nlrp>)
    -->
    (write (crlf) |emia.soar: Elaborate lookup table, dynamic content (sergeant)| (crlf))
    (<nlr> ^sgt |"The sergeant"|)
    (<nlrp> ^sgt |"the sergeant's"|)}

sp {top-ps*emia*elaborate*lookup-table*dynamic-content*self
    (state <s> ^agent-name <name>
       ^emia-template-filling <etf>
       ^name top-state)
    (<etf> ^nl-responsible_agent <nlr>
   ^nl-responsible_agent-poss <nlrp>)
    -->
    (write (crlf) |emia.soar: Elaborate lookup table, dynamic content (self)| (crlf))
    (<nlr> ^<name> |"I am"|)
    (<nlrp> ^<name> |my|)}


###############################################################################
# UNDERSTAND-SPEECH RULES
# ~~~~~~~~~~~~~~~~~~~~~~
#
# These rules scan for certain keywords in order to detect a users question. When
# a rule fires, it will put the semantics of the question in 'emotion-semantics'
# Some rules in the Misc sections will select the right emotion-semantics, when
# more candidates are available.
# These rules are classified by type as can be read in the report 'Interaction
# on Emotion' by Hartholt and Muller.
#
# Some Misc understand-speech rules can be found elsewhere in this file.
#
###############################################################################

##############
# Emotion type
##############


# Scans for 'How do you feel?'
sp {top-state*apply*operator*understand-speech*nlu*how-do-you-feel
  (state <s> ^name top-state
     ^operator <o>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
```

```
  (<i> ^token.lex << how hows >>
       ^token.lex << feel feeling hanging hangin going are >>
       -^token.lex << about >>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem> + & ) ;# & collects all the emotion-semantics, code in the
  (<sem> ^type question       ;# misc section of this file will pick the right one
 ^q-slot value       ;# It is used here because some questions can trigger
 ^prop <sem1>      ;# this rule twice and one instance has to be selected
 ^priority 1)
  (<sem1> ^attribute max-feeling
  ^type state
  ^objec-id <self>)}

# Scans for 'What is wrong with you?'
sp {top-state*apply*operator*understand-speech*nlu*whats-wrong-with-you
  (state <s> ^name top-state
     ^operator <o>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
  (<i> ^token.lex << what whats >>
       ^token.lex << wrong mind >>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem>)
  (<sem> ^type question
 ^q-slot value
 ^prop <sem1>)
  (<sem1> ^attribute max-feeling
  ^type state
  ^objec-id <self>)}

# Scans for 'Do you feel [emotion]?'
sp {top-state*apply*operator*understand-speech*nlu*do-you-feel-emotion
  (state <s> ^name top-state
     ^operator <o>
     ^lexicon <lexicon>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
  (<lexicon> ^emotions.<emotion> <lex>)
  (<i> ^token.lex << feel feeling are seem look >>
       ^token.lex <lex>
       -^token.lex << responsible why what whats who whos whom about >> )
  -->
  (<i> ^mood question
       ^emotion-semantics <sem>)
  (<sem> ^type question
 ^q-slot polarity
 ^prop <sem1>)
  (<sem1> ^attribute feeling
  ^value <emotion>
  ^type state
  ^objec-id <self>)}

# Scans for 'How do you feel about [state]?', one keyword per state
sp {top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*one
  (state <s> ^name top-state
     ^operator <o>
```

```
     ^lexicon <lexicon>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
  (<i> ^token.lex << feel feeling feelings think >>
       ^token.lex << about of >>
       ^token.lex <word>)
  (<lexicon> ^{<domain> <> states <> emotion-responsibility-states <> actions}.<word-int> <word>
     ^states.<state> <word-int>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem> + & ) ;# & collects all the emotion-semantics, code in the
  (<sem> ^type question               ;# misc section of this file will pick the right one
 ^q-slot value
 ^priority 1
 ^prop <sem1>)
  (<sem1> ^attribute feeling-about-state
  ^cause <state>
  ^type state
  ^objec-id <self>)}

# Scans for 'How do you feel about [state]?', two keywords per state
sp {top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*two
  (state <s> ^name top-state
     ^operator <o>
     ^lexicon <lexicon>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
  (<i> ^token.lex << feel feeling feelings think >>
       ^token.lex << about of >>
       ^token.lex <word1>
       ^token.lex {<word2> <> <word1>})
  (<lexicon> ^{<domain1> <> states <> emotion-responsibility-states
                <> actions}.<word-int1> <word1>
     ^{<domain2> <> states <> emotion-responsibility-states
                 <> actions}.<word-int2> <word2>
     ^states.<state> <word-int1>
     ^states.<state> <word-int2>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem> + & )
  (<sem> ^type question
 ^q-slot value
 ^priority 2
 ^prop <sem1>)
  (<sem1> ^attribute feeling-about-state
  ^cause <state>
  ^type state
  ^objec-id <self>)}

# Scans for 'How do you feel about [state]?', three keywords per state
sp {top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*three
  (state <s> ^name top-state
     ^operator <o>
     ^lexicon <lexicon>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
```

```
  (<i> ^token.lex << feel feeling feelings think >>
       ^token.lex << about of >>
       ^token.lex <word1>
       ^token.lex {<word2> <> <word1> <> <word3>}
       ^token.lex {<word3> <> <word1> <> <word2>})
  (<lexicon> ^{<domain1> <> states <> emotion-responsibility-states <> actions}.<word-int1> <word1>
     ^{<domain2> <> states <> emotion-responsibility-states <> actions}.<word-int2> <word2>
     ^{<domain3> <> states <> emotion-responsibility-states <> actions}.<word-int3> <word3>
     ^states.<state> <word-int1>
     ^states.<state> <word-int2>
     ^states.<state> <word-int3>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem> + & )
  (<sem> ^type question
 ^q-slot value
 ^priority 3
 ^prop <sem1>)
  (<sem1> ^attribute feeling-about-state
  ^cause <state>
  ^type state
  ^objec-id <self>)}

###################
# Emotion intensity
###################

# Scans for 'Calm down'
sp {top-state*apply*operator*understand-speech*nlu*calm-down
  (state <s> ^name top-state
     ^operator <o>
     ^lexicon <lexicon>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
  (<i> ^token.lex << calm >>
       ^token.lex << down >>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem>)
  (<sem> ^type question
 ^q-slot dummy-int ;# formal semantics do not match natural semantics, but this
 ^prop <sem1>)     ;# is needed to detect it for the right answer further down the line
  (<sem1> ^attribute max-feeling
  ^type state
  ^objec-id <self>)}

# Scans for 'Relax'
sp {top-state*apply*operator*understand-speech*nlu*relax
  (state <s> ^name top-state
     ^operator <o>
     ^lexicon <lexicon>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
  (<i> ^token.lex << relax >>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem>)
  (<sem> ^type question
```

```
 ^q-slot dummy-int ;# formal semantics do not match natural semantics, but this
 ^prop <sem1>)     ;# is needed to detect it for the right answer further down the line
  (<sem1> ^attribute max-feeling
  ^type state
  ^objec-id <self>)}

###############
# Emotion state
###############

# Scans for 'Why do you feel [emotion]?'
sp {top-state*apply*operator*understand-speech*nlu*why-do-you-feel-emotion
  (state <s> ^name top-state
     ^operator <o>
     ^lexicon <lexicon>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
  (<lexicon> ^emotions.<emotion> <lex>)
  (<i> ^token.lex << why >>
       ^token.lex << feel are >>
       ^token.lex <lex>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem>)
  (<sem> ^type question
 ^q-slot cause
 ^prop <sem1>)
  (<sem1> ^attribute feeling
  ^type state
  ^value <emotion>
  ^objec-id <self>)}

# Scans for 'What's causing you to feel [emotion]?'
sp {top-state*apply*operator*understand-speech*nlu*whats-causing-you-to-feel-emotion
  (state <s> ^name top-state
     ^operator <o>
     ^lexicon <lexicon>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
  (<lexicon> ^emotions.<emotion> <lex>)
  (<i> ^token.lex << what whats causing >>
       ^token.lex << feel be about >>
       ^token.lex <lex>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem> + & )
  (<sem> ^type question
 ^q-slot cause
 ^prop <sem1>
 ^priority 1)
  (<sem1> ^attribute feeling
  ^type state
  ^value <emotion>
  ^objec-id <self>)}

########################
# Emotion responsibility
########################
```

```
# Scans for 'Who's responsible for making you feel [emotion]?'
sp {top-state*apply*operator*understand-speech*nlu*responsible-for-emotion
  (state <s> ^name top-state
     ^operator <o>
     ^lexicon <lexicon>
     ^agent-name <self>)
  (<o> ^name understand-speech
       ^speech-input <si>)
  (<si> ^interpretation <i>)
  (<lexicon> ^emotions.<emotion> <lex>)
  (<i> ^token.lex << who whos whom >>
       ^token.lex << you >>
       ^token.lex <lex>)
  -->
  (<i> ^mood question
       ^emotion-semantics <sem> + & )
  (<sem> ^type question
 ^q-slot value
 ^prop <sem1>
 ^priority 1)
  (<sem1> ^attribute (tcl |stateResponsible | <emotion>)
  ^type state
  ^objec-id <self>)}


##############################################################################
# OUTPUT-SPEECH RULES
# ~~~~~~~~~~~~~~~~~~~
#
# These rules detect the agent's intention to say something about emotions and
# generate the proper output, using templates. These templates are TCL code,
# which can be found elsewhere in this file.
#
##############################################################################

###############
# Emotion type
###############

# Directs the goal associated with the question 'How do you feel'
# and alikes to the proper TCL template
sp {top-state*apply*operator*output-speech*answer-how-do-you-feel*cause
  (state <s> ^agent-name <me>
     ^operator <o>
     ^io.input-link <io>
     ^emia-template-filling <etf>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot value
^prop <p>
^reference <ems>)
  (<p> ^attribute max-feeling)
  (<ems> ^attribute max-feeling ;# You could lose this, as for now there's only one reference which
 ^cause.name <cause>    ;# always has the attribute max-feeling. Same holds for similar rules
```

```
 ^value <emotion>
 ^source.appraisal.type <infl>
 ^source.appraisal.feature.status <st>)
  (<etf> ^nl-emotion.<emotion> <nl_emotion>
 ^nl-intensity-lower <nlil>
 ^nl-intensity-upper <nliu>
 ^nl-influence-status.<infl>.<st> <influence-status>
 ^nl-state-pre.<cause> <state_pre>
 ^nl-state-post.<cause> <state_post>)
  (<nlil> ^<valuelow> <nl_intensity>)
  (<nliu> ^<valueupper> <nl_intensity>)
  (<io> ^|Terseness| <terseness-intensity>
^self <self>)
  (<self> ^<emotion> {<emotion_intensity> >= <valuelow> < <valueupper>})
  -->
  (<cg> ^surface (tcl |emotionTypeFeelCause | <emotion_intensity> | | <terseness-intensity>
     | |<nl_emotion> | | <nl_intensity> | | <state_pre> | | <influence-status> | | <state_post>) + > )}

# Directs the goal associated with the question 'How do you feel'
#and alikes to the proper TCL template
sp {top-state*apply*operator*output-speech*answer-how-do-you-feel*no-cause
  (state <s> ^agent-name <me>
     ^operator <o>
     ^io.input-link <io>
     ^emia-template-filling <etf>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot value
^prop <p>
^reference <ems>)
  (<ems> ^attribute max-feeling
 ^value <emotion>)
  (<etf> ^nl-emotion.<emotion> <nl-emotion>
 ^nl-intensity-lower <nlil>
 ^nl-intensity-upper <nliu>)
  (<nlil> ^<value-low> <nl-intensity>)
  (<nliu> ^<value-upper> <nl-intensity>)
  (<io> ^|Terseness| <terseness-intensity>
^self <self>)
  (<self> ^<emotion> {<emotion-intensity> >= <value-low> < <value-upper>})
  -->
  (<cg> ^surface (tcl |emotionTypeFeelNoCause | <emotion-intensity> | | <terseness-intensity> | | <nl-emo

# Directs the goal associated with the question 'Are you [emotion]'
# and alikes to the proper TCL template
sp {apply*output-speech*answer-do-you-feel-emotion*cause
  (state <s> ^agent-name <me>
     ^operator <o>
     ^io.input-link <io>
     ^emia-template-filling <etf>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
```

```
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot polarity
^prop <p>
^reference <ems>)
  (<p> ^attribute feeling
       ^value <emotion>)
  (<ems> ^attribute feeling
 ^cause.name <cause>
 ^value <emotion>
 ^source.appraisal.type <infl>
 ^source.appraisal.feature.status <st>)
  (<etf> ^nl-influence-status.<infl>.<st> <influence-status>
 ^nl-state-pre.<cause> <state_pre>
 ^nl-state-post.<cause> <state_post>)
  (<io> ^|Terseness| <terseness_intensity>
^self <self>)
  (<self> ^<emotion> {<emotion_intensity> >= 0})
  -->
  (<cg> ^surface (tcl |emotionTypeEmotionCause | <emotion_intensity> | | <terseness_intensity>
  | | <state_pre> | | <influence-status> | | <state_post>) +  > )}

# Directs the goal associated with the question 'Are you [emotion]'
# and alikes to the proper TCL template
sp {apply*output-speech*answer-do-you-feel-emotion*no-cause
  (state <s> ^agent-name <me>
     ^operator <o>
     ^io.input-link <io>
     ^current-state <cs>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot polarity
^prop <p>
^reference <ems>)
  (<p> ^value <emotion>
       ^attribute feeling)
  (<cs> ^<emotion-state> <ems>)
  (<ems> ^attribute feeling
 ^value <emotion>)
  (<io> ^self <self>)
  (<self> ^<emotion> {<emotion_intensity> >= 0})
  -->
  (<cg> ^surface (tcl |emotionTypeEmotionNoCause | <emotion_intensity>))}


# As feeling-about-state states are generated dynamicly, the system does not
# generate references when this type of question is asked. The assertion rules
# for these questions are therefore implemented without the use of references.

# Directs the goal associated with the question 'How do you feel about [state]'
# and alikes to the proper TCL template
sp {top-state*apply*operator*output-speech*answer-emotion-about-state*cause
  (state <s> ^agent-name <me>
```

```
     ^operator <o>
     ^io.input-link <io>
     ^current-state <cs>
     ^emia-template-filling <etf>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot value
^prop <p>)
  (<p> ^attribute feeling-about-state
       ^cause <cause>)
  (<cs> ^<emotion-state> <ems>
^<cause>.belief true)
  (<ems> ^attribute feeling-about-state
 ^cause.name <cause>
 ^value <emotion>
 ^source.appraisal.type <infl>
 ^source.appraisal.feature.status <st>)
  (<etf> ^nl-emotion.<emotion> <nl-emotion>
 ^nl-intensity-lower <nlil>
 ^nl-intensity-upper <nliu>
 ^nl-influence-status.<infl>.<st> <influence-status>
 ^nl-state-pre.<cause> <state-pre>)
  (<nlil> ^<value-low> <nl-intensity>)
  (<nliu> ^<value-upper> <nl-intensity>)
  (<io> ^|Terseness| <terseness-intensity>
^self <self>)
  (<self> ^<emotion> {<emotion-intensity> >= <value-low> < <value-upper>})
  -->
 (<cg> ^surface (tcl |emotionTypeState | <terseness-intensity> | | <nl-intensity>
   | | <nl-emotion> | | <state-pre>) + > )}

# Directs the goal associated with the question 'How do you feel about [state]]'
# and alikes to a surface
sp {top-state*apply*operator*output-speech*answer-emotion-about-state*no-cause
  (state <s> ^agent-name <me>
     ^operator <o>
     ^io.input-link <io>
     ^emia-template-filling <etf>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot value
^prop <p>)
  (<p> ^attribute feeling-about-state)
  -->
  (<cg> ^surface |"That's not an issue right now sir"|)}

###################
# Emotion intensity
```

```
###################

# Directs the goal associated with the utterance 'calm down'
# and alikes to the proper TCL template
sp {top-state*apply*operator*output-speech*answer-calm-down
  (state <s> ^agent-name <me>
     ^operator <o>
     ^io.input-link <io>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot dummy-int
^prop <p>
^reference <ems>)
  (<p> ^attribute max-feeling)
  (<ems> ^attribute max-feeling
 ^value <emotion>)
  (<io> ^self <self>)
  (<self> ^<emotion> <emotion_intensity>)
  -->
  (<cg> ^surface (tcl |emotionCalm | <emotion_intensity>) + = )}

###############
# Emotion state
###############

# Directs the goal associated with the question 'Why are you [emotion]'
# and alikes to the proper TCL template
sp {apply*output-speech*answer-why-do-you-feel-emotion*cause
  (state <s> ^agent-name <me>
     ^operator <o>
     ^io.input-link <io>
     ^emia-template-filling <etf>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot cause
^prop <p>
^reference <ems>)
  (<p> ^attribute feeling
       ^value <emotion>)
  (<ems> ^attribute feeling
 ^cause.name <cause>
 ^value <emotion>
 ^source.appraisal.type <infl>
 ^source.appraisal.feature.status <st>
 ^source.appraisal.evaluation.evaluation <blameworthiness>
 ^source.appraisal.evaluation.responsible-agent <responsible-agent>)
  (<etf> ^nl-emotion.<emotion> <nl-emotion>
 ^nl-influence-status.<infl>.<st> <influence-status>
```

```
 ^nl-state-pre.<cause> <state-pre>
 ^nl-state-post.<cause> <state-post>
 ^nl-responsible_agent-poss.<responsible-agent> <responsible-agent-poss>)
  (<io> ^|Defensiveness| <defensiveness-intensity>
^|Terseness| <terseness-intensity>
^self <self>)
  (<self> ^<emotion> {<emotion-intensity> >= 0})
  -->
  (<cg> ^surface-candidate <sc> + & )
  (<sc> ^output (tcl |emotionWhyCause | <emotion-intensity> | | <terseness-intensity>
    | | <defensiveness-intensity> | | <nl-emotion> | | <emotion> | | <state-pre>
    | | <influence-status> | | <state-post> | | <responsible-agent-poss>
    | | <blameworthiness>)
^priority 3)}

# Directs the goal associated with the question 'Why are you [emotion]'
# and alikes to the proper TCL template
sp {apply*output-speech*answer-why-do-you-feel-emotion*no-responsible-agent
  (state <s> ^agent-name <me>
     ^operator <o>
     ^io.input-link <io>
     ^emia-template-filling <etf>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot cause
^prop <p>
^reference <ems>)
  (<p> ^attribute feeling
       ^value <emotion>)
  (<ems> ^attribute feeling
 ^cause.name <cause>
 ^value <emotion>
 ^source.appraisal.type <infl>
 ^source.appraisal.feature.status <st>)
  (<etf> ^nl-emotion.<emotion> <nl-emotion>
 ^nl-influence-status.<infl>.<st> <influence-status>
 ^nl-state-pre.<cause> <state-pre>
 ^nl-state-post.<cause> <state-post>)
  (<io> ^|Defensiveness| <defensiveness-intensity>
^|Terseness| <terseness-intensity>
^self <self>)
  (<self> ^<emotion> {<emotion-intensity> >= 0})
  -->
  (<cg> ^surface-candidate <sc> + &)
  (<sc> ^output (tcl |emotionWhyNoResponsibleAgent | <emotion-intensity>
    | | <terseness-intensity> | | <defensiveness-intensity> | | <nl-emotion>
    | | <state-pre> | | <influence-status> | | <state-post>)
^priority 2)}

# Directs the goal associated with the question 'Why are you [emotion]'
# and alikes to the proper TCL template
sp {apply*output-speech*answer-why-do-you-feel-emotion*no-cause
  (state <s> ^agent-name <me>
     ^operator <o>
     ^io.input-link <io>
```

```
     ^emia-template-filling <etf>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
       ^q-slot cause
       ^prop <p>
       ^reference <ems>)
  (<p> ^value <emotion>
       ^attribute feeling)
  (<ems> ^attribute feeling
 ^value <emotion>)
  (<etf> ^nl-emotion.<emotion> <nl-emotion>)
  (<io> ^self <self>)
  (<self> ^<emotion> {<emotion-intensity> >= 0})
  -->
  (<cg> ^surface-candidate <sc> + &)
  (<sc> ^output (tcl |emotionWhyNoCause | <emotion-intensity> | | <nl-emotion>)
^priority 1)}

########################
# Emotion responsibility
########################

# Directs the goal associated with the question 'Who's resonsible for making you [emotion]'
# and alikes to the proper TCL template
sp {apply*output-speech*answer-whos-responsible*emotion
  (state <s> ^agent-name <me>
     ^operator <o>
     ^current-state <cs>
     ^io.input-link <io>
     ^emia-template-filling <etf>
     ^lexicon <l>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot value
^prop <p>
^reference <ems>)
  (<p> ^attribute <responsible-for-emotion>)
  (<l> ^emotion-responsibility-states.<responsible-for-emotion> <lex>
       ^emotions.<emotion> <lex>)
  (<ems> ^attribute <responsible-for-emotion>
 ^cause <blameworthiness>
 ^value <agent>)
  (<etf> ^nl-emotion.<emotion> <nl-emotion>
 ^nl-responsible_agent.<agent> <responsible-agent>
 ^nl-responsible_agent-poss.<agent> <responsible-agent-poss>)
  (<io> ^|Terseness| <terseness-intensity>
^self <self>)
  (<self> ^<emotion> <emotion-intensity>)
```

```
  -->
  (<cg> ^surface (tcl |emotionResponsibility | <terseness-intensity>
  | | <emotion-intensity> | | <nl-emotion> | | <emotion> | | <responsible-agent>
  | | <responsible-agent-poss> | | <blameworthiness>) + > )}

# Directs the goal associated with the question 'Who's resonsible for making you [emotion]'
# and alikes to the proper TCL template
sp {apply*output-speech*answer-whos-responsible*emotion*no-responsible-agent
  (state <s> ^agent-name <me>
     ^operator <o>
     ^current-state <cs>
     ^lexicon <l>
     ^io.input-link <io>
     ^emia-template-filling <etf>)
  (<o> ^name output-speech
       ^comgoal <cg>
       ^goal <g>)
  (<g> ^content <c>
       ^action address
       ^addressee lt)
  (<c> ^content <ct>
       ^action info-req)
  (<ct> ^type question
^q-slot value
^prop <p>)
  (<p> ^attribute <responsible-for-emotion>)
  (<l> ^emotion-responsibility-states.<responsible-for-emotion> <lex>
       ^emotions.<emotion> <lex>)
  (<etf> ^nl-emotion.<emotion> <nl-emotion>)
  (<io> ^self.<emotion> <emotion-intensity>)
  -->
  (<cg> ^surface (tcl |emotionNoResponsibility | <emotion-intensity> | | <nl-emotion>))}


################################################################################
# TEMPLATES
# ~~~~~~~~~
#
# These TCL procedures are called by Soar rules passing the appropiate
# information in order to construct a natural language utterance.
#
################################################################################


##############
# Emotion type
##############

# Template for answering questions like 'How do you feel?'
proc emotionTypeFeelCause {emotionIntensityNumber tersenessIntensity emotion
emotionIntensity statePre influenceStatus statePost} {

  set utterance "\"I'm "

  if {$emotionIntensityNumber <= 0.2} {
    append utterance "fine sir\""
  } elseif {$tersenessIntensity < 0.2} {
    append utterance "feeling " $emotionIntensity " " $emotion " because " $statePre
    " " $influenceStatus " " $statePost\"
  } elseif {$tersenessIntensity < 0.6} {
    append utterance "feeling " $emotionIntensity " " $emotion\"
  } else {
```

```
    append utterance "feeling " $emotion\"
  }

  return $utterance
}

# Template for answering questions like 'How do you feel?'
proc emotionTypeFeelNoCause {emotionIntensityNumber tersenessIntensity emotion
emotionIntensity} {

  set utterance "\"I'm "

  if {$emotionIntensityNumber <= 0.2} {
    append utterance "fine sir\""
  } elseif {$tersenessIntensity < 0.6 } {
    append utterance "feeling " $emotionIntensity " " $emotion\"
  } else {
    append utterance "feeling " $emotion\"
  }

  return $utterance
}

# Template for answering questions like 'Are you feeling [emotion]?'
proc emotionTypeEmotionCause {emotionIntensityNumber tersenessIntensity statePre
influenceStatus statePost} {

  set utterance "\""

  if {$emotionIntensityNumber <= 0.1} {
    append utterance "Not at all sir\""
  } elseif {$emotionIntensityNumber <= 0.2} {
    append utterance "No sir\""
  } elseif {$tersenessIntensity <= 0.25} {
    append utterance "Yes sir because " $statePre " " $influenceStatus " " $statePost\"
  } elseif {$emotionIntensityNumber <= 0.5} {
    append utterance "Yes sir\""
  } else {
    append utterance "Quite a bit sir\""
  }

  return $utterance
}

# Template for answering questions like 'Are you feeling [emotion]?'
proc emotionTypeEmotionNoCause {emotionIntensityNumber} {

  set utterance "\""

  if {$emotionIntensityNumber <= 0.1} {
    append utterance "Not at all sir\""
  } elseif {$emotionIntensityNumber <= 0.2} {
    append utterance "No sir\""
  } elseif {$emotionIntensityNumber <= 0.5} {
    append utterance "Yes sir\""
  } else {
    append utterance "Quite a bit sir\""
  }

  return $utterance
}
```

```
# Template for answering questions like 'How do you feel about [state]?'
proc emotionTypeState {tersenessIntensity emotionIntensity emotion statePre} {

  set utterance "\"I'm "

  if {$tersenessIntensity <= 0.5} {
    append utterance $emotionIntensity " "
  }

  if {$statePre == "the boy" || $statePre == "the driver"} {
    append utterance "feeling " $emotion " about him\""
  } else {
    append utterance "feeling " $emotion " about it sir\""
  }

  return $utterance
}


####################
# Emotion intensity
####################

# Template for reacting to 'Calm down' and alikes
proc emotionCalm {emotionIntensity} {

  set utterance "\""

  if {$emotionIntensity > 0.2} {
    append utterance "Will do sir\""
  } else {
    append utterance "I am calm sir\""
  }

  return $utterance
}

###############
# Emotion state
###############

# Template for answering questions like 'Why do you feel [emotion]?'
proc emotionWhyCause {emotionIntensityNumber tersenessIntensity defensivenessIntensity
emotionNL emotion statePre influenceStatus statePost responsibleAgentPoss blameworthiness} {

  set utterance "\""

  if {$emotionIntensityNumber <= 0.2} {
    append utterance "I'm not feeling " $emotionNL " sir\""
  } elseif {$defensivenessIntensity > 0.75} {
    append utterance "That's personal sir\""
  } elseif {$blameworthiness == "blameworthy" && $emotion != "Hope" && $emotion != "Joy"} {
    append utterance $statePre " " $influenceStatus " " $statePost " sir that's "
    $responsibleAgentPoss " fault\""
  } else {
    append utterance $statePre " " $influenceStatus " " $statePost " sir\""
  }

  return $utterance
}

# Template for answering questions like 'Why do you feel [emotion]?'
```

```
proc emotionWhyNoResponsibleAgent {emotionIntensityNumber tersenessIntensity
defensivenessIntensity emotion statePre influenceStatus statePost} {

  set utterance "\""

  if {$emotionIntensityNumber <= 0.2} {
    append utterance "I'm not feeling " $emotion " sir\""
  } elseif {$defensivenessIntensity > 0.75} {
    append utterance "That's personal sir\""
  } else {
    append utterance $statePre " " $influenceStatus " " $statePost " sir\""
  }

  return $utterance
}

# Template for answering questions like 'Why do you feel [emotion]?'
proc emotionWhyNoCause {emotionIntensityNumber emotion} {

  set utterance "\""

  if {$emotionIntensityNumber <= 0.1} {
    append utterance "I'm not feeling " $emotion " sir\""
  } else {
    append utterance "I think because you are playing with my sliders
    Please quit it   It's pretty annoying te be thrown between emotions like that\""
  }

  return $utterance
}

########################
# Emotion responsibility
########################

# Template for answering questions like 'Who's responsible for making you feel [emotion]?'
proc emotionResponsibility {tersenessIntensity emotionIntensity emotionNL emotion
responsibleAgent responsibleAgentPoss blameworthiness} {

  set utterance "\""

  if {$emotionIntensity <= 0.2} {
    append utterance "I'm not feeling " $emotionNL " sir\""
  } elseif {$tersenessIntensity <= 0.25 && $blameworthiness == "blameworthy" &&
    $emotion != "Hope" && $emotion != "Joy"} {
    append utterance "That's " $responsibleAgentPoss " fault sir\""
  } else {
    append utterance $responsibleAgent " sir\""
  }

  return $utterance
}

# Template for answering questions like 'Who's responsible for making you feel [emotion]?'
proc emotionNoResponsibility {emotionIntensity emotionNL} {

  set utterance "\""

  if {$emotionIntensity <= 0.2} {
    append utterance "I'm not feeling " $emotionNL " sir\""
  } else {
    append utterance "No one in particular sir\""
```

```
  }

  return $utterance
}

##############################################################################
# MISCELLANEOUS RULES
# ~~~~~~~~~~~~~~~~~~
#
##############################################################################

# If there are two emotion-semantics, the one with the highest priority is preferred
sp {top-state*apply*operator*understand-speech*emotion-semantics-selection*priority
  (state <s> ^name top-state
    ^operator <o>)
  (<o> ^name understand-speech
       ^speech-input.interpretation <i>)
  (<i> ^emotion-semantics <sem1>
       ^emotion-semantics <sem2>)
  (<sem1> ^priority <p1>)
  (<sem2> ^priority {<p2> < <p1>})
  -->
  (<i> ^emotion-semantics <sem1> > <sem2>
       ^emotion-semantics <sem1> + )}

# If there are two emotion-semantics with the same priority, just pick one
sp {top-state*apply*operator*understand-speech*emotion-semantics-selection*same
  (state <s> ^name top-state
    ^operator <o>)
  (<o> ^name understand-speech
       ^speech-input.interpretation <i>)
  (<i> ^emotion-semantics <sem1>
       ^emotion-semantics {<sem2> <> <sem1>})
  (<sem1> ^priority <p1>)
  (<sem2> ^priority <p1>)
  -->
  (<i> ^emotion-semantics <sem1> + = )}

# If there are two emotion-semantics, and one of them has an associated emotion with
# a cause and the other one an emotion without a cause, the one with the cause is preffered
# the one with the cause is preffered
sp {top-state*apply*operator*understand-speech*emotion-semantics-selection*
cause-vs-rebel-without-a-cause
  (state <s> ^name top-state
     ^operator <o>
     ^appraisals <appr>)
  (<o> ^name understand-speech
       ^speech-input.interpretation <i>)
  (<i> ^emotion-semantics <sem1>
       ^emotion-semantics {<sem2> <> <sem1>})
  (<sem1> ^priority <p1>
  ^prop.cause <state1>)
  (<sem2> ^priority <p1>
  ^prop.cause <state2>)
  (<appr> ^<emotion1>.appraisal.object.name <state1>
  -^<emotion2>.appraisal.object.name <state2>)
  -->
  (<i> ^emotion-semantics <sem1> > <sem2>
       ^emotion-semantics <sem1> + )}

# If there are two emotion-semantics, and one of them has an associated emotion with an
# intensity and the other one an emotion without an intensity, the one with the intensity
```

```
# is preffered
sp {top-state*apply*operator*understand-speech*emotion-semantics-selection*
intensity-vs-no-intensity
  (state <s> ^name top-state
     ^operator <o>
     ^current-state <cs>
     ^appraisals <appr>)
  (<o> ^name understand-speech
       ^speech-input.interpretation <i>)
  (<i> ^emotion-semantics <sem1>
       ^emotion-semantics {<sem2> <> <sem1>})
  (<sem1> ^priority <p1>
  ^prop.cause <state1>)
  (<sem2> ^priority <p1>
  ^prop.cause <state2>)
  (<cs> ^<state1>.emotion.intensity <dummy1>
^<state2>.emotion <emotion2>)
  -(<emotion2> ^intensity <dummy2>)
  -->
  (<i> ^emotion-semantics <sem1> > <sem2>
       ^emotion-semantics <sem1> + = )}

# If both the NLU and the emotion code propose semantics, the emotion semantics is preferred
sp {top-state*apply*operator*understand-speech*emotion-semantics-preference*sem
  (state <s> ^name top-state
     ^operator <o>)
  (<o> ^name understand-speech
       ^speech-input.interpretation <i>)
  (<i> ^emotion-semantics <sem>
       -^emotion-semantics {<sem2> <> <sem>}
       ^nlu-interp.sem <sem3>)
-->
  (<i> ^semantics <sem> > <sem3>
       ^semantics <sem> + ) }

# If both the NLU and the emotion code propose semantics, the emotion semantics is preferred
sp {top-state*apply*operator*understand-speech*emotion-semantics-preference*semantics
  (state <s> ^name top-state
     ^operator <o>)
  (<o> ^name understand-speech
       ^speech-input.interpretation <i>)
  (<i> ^emotion-semantics <sem>
       -^emotion-semantics {<sem2> <> <sem>}
       ^nlu-interp.semantics <sem3>)
-->
  (<i> ^semantics <sem> > <sem3>
       ^semantics <sem> + )}

# If there's a comgoal with a surface attribute (probably caused by emotion code),
# give it a high preference, otherwise, it might lose from whatever the NLU comes up with
sp {top-state*apply*operator*output-speech*comgoal-surface
  (state <s> ^name top-state
     ^operator <o>)
  (<o> ^name output-speech
       ^goal <cg>)
  (<cg> ^surface)
  -->
  (<o> ^comgoal <cg> + >)}

# If there are two speech acts and one of them is incomplete (NLU), the
# other one (emotion speech-act) is better
sp {top-state*apply*operator*output-speech*emotion-speech-act
```

```
  (state <s> ^name top-state
     ^operator <o>)
  (<o> ^name understand-speech
       ^speech-input.interpretation <i>)
  (<i> ^speech-act <sa1>
       ^speech-act {<sa2> <> <sa1>})
  (<sa2> ^content.incomplete yes)
  -->
  (<i> ^speech-act <sa1> + >
       ^speech-act <sa1> > <sa2>)}

# If there are two surface-candidates, the one with the highest priority is preferred
sp {top-state*apply*operator*output-speech*surface-selection*priority
  (state <s> ^name top-state
    ^operator <o>)
  (<o> ^name output-speech
       ^comgoal <cg>)
  (<cg> ^surface-candidate <sc1>
^surface-candidate <sc2>)
  (<sc1> ^priority <p1>
 ^output <o1>)
  (<sc2> ^priority {<p2> < <p1>}
 ^output <o2>)
  -->
  (<cg> ^surface <o1> > <o2>
^surface <o1> + > )}

# If there's one surface-candidate, make it the surface
sp {top-state*apply*operator*output-speech*surface-selection*copy
  (state <s> ^name top-state
    ^operator <o>)
  (<o> ^name output-speech
       ^comgoal <cg>)
  (<cg> ^surface-candidate <sc1>
-^surface-candidate {<sc2> <> <sc1>})
  (<sc1> ^output <o1>)
  -->
  (<cg> ^surface <o1> + > )}

# TCL prodedure which produces the attribute name for the responsible agent question
proc stateResponsible {emotion} {

  set utterance responsible-for-
  append utterance $emotion
  return $utterance
}

# Adds attribute '^time future' to sem, when a state has nog happened yet. Necessary
# to get the right tense in the answer when asking 'why' as a follow up question to for
# instance ``I'm hopeful''.
 sp {top-state*apply*operator*output-speech*answer-state*why-question*assert*speech-acts*
 emotion-future
  (state <s> ^name top-state
     ^operator <o>
     ^agent-name <me>)
  (<o> ^name output-speech
       ^goal <obl>
       ^comgoal <cg>)
  (<obl> ^action address
 ^content <csa>)
  (<csa> ^action info-req
 ^content <sem>)
```

```
 (<sem> ^type question
^prop <sem1>
^q-slot cause
^reference  <ref>)
 (<ref> ^attribute << max-feeling feeling >>
^cause <cause>
^source.appraisal.feature.status unconfirmed)
 (<sem1> ^type state)
 (<cg> ^sem <semcg>)
 -->
 (write (crlf) TEST (crlf))
 (<semcg> ^time future + > )}
```

B.3 Lexicon-emia.soar

```
### Copyright 2003 Arno Hartholt  |Tijmen Joppe Muller
### Institute for Creative Technologies
##############################################################################
###
### File              : lexicon-emia.soar
### Original author(s): Arno Hartholt (hartholt@ict.usc.edu)
###                     Tijmen Joppe Muller (tijmen@avpec1910.nl)
### Supervisor        : Jonathan Gratch (gratch@ict.usc.edu)
### Organization      : Institute for Creative Technologies
### Created on        : November 13, 2003
### Last Modified By  : Arno Hartholt
### Soar Version      : 7
### Documentation     : Interaction on emotions
###                     (http://mullert.adsl.utwente.nl/~stage/reports/emia.pdf)
###____________________________________________________________________________
### HISTORY
###
### 11-17-03 [AH]   Document created, only emotions
### 11-26-03 [AH]   Added <states> and <emotion-responsibility-states>
### 12-04-03 [AH]   Added <misc>
### 12-29-03 [AH]   Added new states and some misc words
### 01-02-04 [AH]   Added synonym 'scared'
###
###____________________________________________________________________________
### USE
###
### Serves as an add-on for lexicon.soar.
### <states> sums up all states and the words used to point to a certain state.
### <emotions> is used to get to the internal representation of an emotion from
###            natural language
### <emotion-responsibility-states> is not really used for mapping of natural language
###                                 to internal representation, but for detecting
###                                 the right goal in output-speech
### <misc> is a list of words which could be included in one of the original lexicon
###        entries
###____________________________________________________________________________
### ISSUES
###
### As both the words 'mark' and 'secure' point to the secure attribute in the
### original lexicon, this can lead to a misunderstanding of the user, as the
### distinction between lz-marked and lz-secure cannot always be made.
###____________________________________________________________________________
### KNOWN BUGS
###
```

```
###_____________________________________________________________________________
### LIMITATIONS
###
###_____________________________________________________________________________
### EXTERNAL REFERENCES
###
### Used in combination with emotion-states-emia.soar and language-emia.soar
###_____________________________________________________________________________
### TO DO
###
###_____________________________________________________________________________
### SUMMARY
###
###
###############################################################################

echo "\nLoading lexicon-emia.soar\n"

# Adds words concerning emotions to the allready existing lexicon
sp {top-state*elaborate*state*add-lexicon-entries*emotions
  (state <s> ^name top-state
     ^agent-name << sgt medic director >>
     ^lexicon <lexicon>)
-->
  (<lexicon> ^states <states>
     ^emotions <emotions>
     ^emotion-responsibility-states <emotion-responsibility-states>
     ^misc <misc>)
  (<states> ^maintain-goodwill crowd + &, goodwill + &
    ^crowd-angry crowd + &, |Anger| + &
    ^boy-dead boy + &, dead + &
    ^boy-healthy boy + &, healthy + &
    ^minor-injuries minor-injuries + &, injuries + &, boy + &
    ^serious-injuries serious-injuries + &, injuries + &, boy + &
    ^critical-injuries critical-injuries + &, injuries + &, boy + &
    ^driver-healthy driver + &, healthy + &
    ^driver-minor-inj driver + &, minor-injuries + &, injuries + &
    ^mother-healthy mom + &, healthy + &
    ^know-boy-health boy + &, healthy + &, known + &
    ^sqds-in-transit squads + &, transit + &
    ^lt-in-transit lt + &, transit + &
    ^lt-at-aa lt  + &, aa + &
    ^lt-at-celic lt + &, celic + &
    ^sgt-at-aa sgt + &, aa + &
    ^medic-at-aa medic + &, aa + &
    ^at-boy-aa boy + &, aa + &
    ^1st-sqd-at-aa 1st-sqd + &, aa + &
    ^1st-sqd-at-celic 1st-sqd + &, celic + &
    ^1st-sqd-at-lz 1st-sqd + &, lz + &
    ^1st-sqd-in-transit 1st-sqd + &, transit + &
    ^2nd-sqd-at-aa 2nd-sqd + &, aa + &
    ^2nd-sqd-at-celic 2nd-sqd + &, celic + &
    ^3rd-sqd-at-aa 3rd-sqd + &, aa + &
    ^3rd-sqd-at-celic 3rd-sqd + &, celic + &
    ^3rd-sqd-at-lz 3rd-sqd + &, lz + &
    ^4th-sqd-at-aa 4th-sqd + &, aa + &
    ^4th-sqd-at-celic 4th-sqd + &, celic + &
    ^4th-sqd-at-lz 4th-sqd + &, lz + &
    ^4th-sqd-in-transit 4th-sqd + &, transit + &
    ^at-boy-hospital boy + &, hospital + &
    ^mom-in-intersect mom + &, intersection + &
    ^sqd-in-intersect sqd + &, intersection + &
```

```
  ^medevac-at-aa medevac + &, aa + &
  ^medevac-at-base medevac + &, base + &
  ^medevac-called medevac + &, called + &
  ^medevac-overhead medevac + &, overhead + &
  ^amb-at-aa amb + &, aa + &
  ^amb-at-base amb + &, base + &
  ^ambulance-called amb + &, called + &
  ^secure-route route + &, secured + &
  ^aa-secure aa + &, secured + &
  ^accident-secure accident + &, secured  + &
  ^12-to-4-secure 12-to-4 + &, secured + &
  ^4-to-8-secure 4-to-8 + &, secured + &
  ^8-to-12-secure 8-to-12 + &, secured + &
  ^lz-secure lz + &, secured + &
  ^lz-marked lz + &, marked + &
  ^lz-clear lz + &, clear + &
  ^support-1-6 eagle2-6 + &, support + &
  ^retain-mass eagle2-6 + &, together + &
  ^fracture-unit eagle2-6 + &, fractured + &
  ^1-6-at-celic eagle1-6 + &, celic + &
  ^hospital-at-tuzla hospital + &, tuzla + &)
(<emotions> ^|Anger| anger + &, angry + &, annoyed + &, irritated + &, pissed + &, mad + &
    ^|Anxiety| anxiety + &, anxious + &
    ^|Distress| distress + &, distressed + &, upset + &, disturbed + &,
     troubled + &, worried + &, concerned + &, sad & +
   ^|Fear| fear + &, fearful + &, afraid + &, frightened + &, stressed + &,
    pessimistic + &, scared + &
   ^|Guilt| guilt + &, guilty + &
   ^|Hope| hope + &, hopeful + &, optimistic + &
   ^|Joy| joy + &, joyful + &, excited + &, happy + &)
(<emotion-responsibility-states> ^|responsible-for-Anger| anger + &, angry + &, annoyed + &,
                                 irritated + &
 ^|responsible-for-Anxiety| anxiety + &, anxious + &, worried + &,
  concerned + &
 ^|responsible-for-Distress| distress + &, distressed + &, upset + &,
  disturbed + &, troubled + &
 ^|responsible-for-Fear| fear + &, fearfull + &, afraid + &, frightened + &,
  stressed + &, pessimistic + &, scared + &
 ^|responsible-for-Guilt| guilt + &, guilty + &
 ^|responsible-for-Hope| hope + &, hopefull + &, optimistic + &
 ^|responsible-for-Joy| joy + &, joyfull + &, exited + &, happy + &)
(<misc> ^crowd crowd + &, bosnians + &
^goodwill goodwill + &
^known known + &, know + &
^intersection intersection + &
^together together + &
^sqd squad + &, sqd + &
^fractured fracture + &, fractured + &, split + &, splitting + &
^eagle1-6 eagle1-6 + &
^injuries injuries + &, injured + &
^call called + &)}
```

Appendix C

Test plans

The statements between < and > refer to unknown identifier names, they are randomly choosen by the Soar system. It is tried to choose them in a way it is clear what complex object's they refer to.

C.1 Phase 1

C.1.1 `top-ps*emia*elaborate*lookup-table*template`

Conditions

None.

Test plan

1. Add rule to the system.

Expected results

1. Creation of WME (`S1 ^emia-template-filling <emia-template-filling>`).
2. Creation of WME (`<emia-template-filling> ^nl-emotion <nl-emotion>`) (equivalent for the other eight `^nl-` attributes).

Results

All expected results were met.

C.1.2 `top-ps*emia*elaborate*emotion-state*max-feeling`

Conditions

1. Initialized agent sergeant is used as testing environment; the intensity for hope is 0.57, the intensity for distress is 0.56 and the intensity for the other emotions are about 0.

Test plan

1. Add rule to the system.
2. Bring intensity of emotion hope below the intensity of distress.
3. Bring intensity of emotion hope equal to the intensity of distress.

Expected results

After first step of the test plan:

1. Creation of WME (<current-state> ^<new1> <newname1>).
2. Creation of WME (<current-state> ^<new2> <newname2>).
3. Creation of WME (<newname1> ^attribute max-feeling).
4. Creation of WME (<newname1> ^belief true).
5. Creation of WME (<newname1> ^polarity positive).
6. Creation of WME (<newname1> ^value |Hope|).
7. Creation of trivial WME's concerning object <newname1>.
8. Creation of object <newname2>, the inverse of <newname1>.

After second step of the test plan:

1. Rejection of object <newname1>.
2. Rejection of object <newname2>.
3. Creation of WME (<current-state> ^<new3> <newname3>).
4. Creation of WME (<current-state> ^<new4> <newname4>).
5. Creation of WME (<newname3> ^attribute max-feeling).
6. Creation of WME (<newname3> ^belief true).
7. Creation of WME (<newname3> ^polarity positive).
8. Creation of WME (<newname3> ^value |Distress|).
9. Creation of trivial WME's concerning object <newname3>.
10. Creation of object <newname4>, the inverse of <newname3>.

After third step of the test plan:

1. Rejection of object <newname3>.
2. Rejection of object <newname4>.
3. Creation of WME (<current-state> ^<new5> <newname5>).
4. Creation of WME (<current-state> ^<new6> <newname6>).

5. Creation of WME (<newname5> ^attribute max-feeling).
6. Creation of WME (<newname5> ^belief true).
7. Creation of WME (<newname5> ^polarity positive).
8. Creation of WME (<newname5> ^value *emotion*), where *emotion* is a random pick between |Hope| and |Distress|.
9. Creation of trivial WME's concerning object <newname5>.
10. Creation of object <newname6>, the inverse of <newname5>.

Results

All expected results were met.

C.1.3 top-ps*emia*elaborate*emotion-state*feeling*true

Conditions

1. Initialized agent sergeant is used as testing environment; the intensity for hope is 0.57, the intensity for distress is 0.56 and the intensity for the other emotions are about 0.

Test plan

1. Add rule to the system.
2. Bring intensity of emotion distress to about 0.
3. Bring intensity of emotion fear to about 0.5.

Expected results

After first step of the test plan:

1. Creation of WME (<current-state> ^<new1> <newname1>).
2. Creation of WME (<current-state> ^<new2> <newname2>).
3. Creation of WME (<newname1> ^attribute feeling).
4. Creation of WME (<newname1> ^belief true).
5. Creation of WME (<newname1> ^polarity positive).
6. Creation of WME (<newname1> ^value |Distress|).
7. Creation of trivial WME's concerning object <newname1>.
8. Creation of object <newname2>, the inverse of <newname1>.
9. Creation of object, equal to <newname1>, but with ^value |Hope|.
10. *No creation* of any other object with ^attribute feeling.

After second step of the test plan:

1. Rejection of object `<newname1>`.
2. Rejection of object `<newname2>`.

After third step of the test plan:

1. Creation of WME `(<current-state> ^<new3> <newname3>)`.
2. Creation of WME `(<current-state> ^<new4> <newname4>)`.
3. Creation of WME `(<newname3> ^attribute feeling)`.
4. Creation of WME `(<newname3> ^belief true)`.
5. Creation of WME `(<newname3> ^polarity positive)`.
6. Creation of WME `(<newname3> ^value |Fear|)`.
7. Creation of trivial WME's concerning object `<newname3>`.
8. Creation of object `<newname4>`, the inverse of `<newname3>`.

Since the design for all seven emotions in the system are equal, we can safely assume that if the above tests are executed correctly, the creation of emotion states for all emotions are covered by this rule.

Results

All expected results were met.

C.1.4 `top-ps*emia*elaborate*emotion-state*feeling*false`

Conditions

1. Initialized agent sergeant is used as testing environment; the intensity for hope is 0.57, the intensity for distress is 0.56 and the intensity for the other emotions are about 0.

Test plan

1. Add rule to the system.
2. Bring intensity of emotion distress to about 0.
3. Bring intensity of emotion fear to about 0.5.

Expected results

After first step of the test plan:

1. Creation of WME (<current-state> ^<new1> <newname1>).
2. Creation of WME (<current-state> ^<new2> <newname2>).
3. Creation of WME (<newname1> ^attribute feeling).
4. Creation of WME (<newname1> ^belief true).
5. Creation of WME (<newname1> ^polarity negative).
6. Creation of WME (<newname1> ^value |Fear|).
7. Creation of trivial WME's concerning object <newname1>.
8. Creation of object <newname2>, the inverse of <newname1>.
9. Creation of objects, equal to <newname1>, but with ^value |Anger|, ^value |Anxiety|, ^value |Guilt|, ^value |Joy|.
10. *No creation* of any other object with ^attribute feeling.

After second step of the test plan:

1. Creation of WME (<current-state> ^<new3> <newname3>).
2. Creation of WME (<current-state> ^<new4> <newname4>).
3. Creation of WME (<newname3> ^attribute feeling).
4. Creation of WME (<newname3> ^belief true).
5. Creation of WME (<newname3> ^polarity negative).
6. Creation of WME (<newname3> ^value |Distress|).
7. Creation of trivial WME's concerning object <newname3>.
8. Creation of object <newname4>, the inverse of <newname3>.

After third step of the test plan:

1. Rejection of object <newname1>.
2. Rejection of object <newname2>.

Since the design for all seven emotions in the system are equal, we can safely assume that if the above tests are executed correctly, the creation of emotion states for all emotions are covered by this rule.

Results

All expected results were met.

C.1.5 top-state*apply*operator*understand-speech*nlu*how-do-you-feel

Conditions

none

Test plan

1. Add rule to the system.
2. Ask "How do you feel?".
3. Ask "How do you feel about the boy?".

Expected results

After first step of test plan: rule should not fire. After second step of test plan:

1. Creation of WME `(^interpretation mood question)`
2. Creation of WME `(^interpretation emotion-semantics <emotion-semantics>)`
3. Creation of WME `(<emotion-semantics> ^type question)`
4. Creation of WME `(<emotion-semantics> ^q-slot value)`
5. Creation of WME `(<emotion-semantics> ^prop <sem>)`
6. Creation of WME `(<sem> ^attribute max-feeling)`
7. Creation of WME `(<sem> ^type state)`

After third step of test plan: rule should not fire

Since the design of this rule is identical to that of `...*nlu*whats-wrong-with-you`, `...*nlu*do-you-feel-emotion`, `...*nlu*calm-down` and `...*nlu*relax`, we can safely assume that if the above tests are executed correctly, all four rules are covered.

Results

All expected results were met.

C.1.6 top-state*elaborate*state*add-lexicon-entries*emotions

Conditions

none

Test plan

1. Add rule to the system.

Expected results

After first step of test plan:

1. Creation of WME `(^lexicon states <states>)`
2. Creation of WME `(^lexicon emotions <emotions>)`
3. Creation of WME `(^lexicon emotion-responsibility-states <emotion-responsibility-states>)`
4. Creation of WME `(^lexicon misc <misc>)`
5. Creation of WME `(<states> ^maintain-goodwill crowd)` (equivalent for other 50 states)
6. Creation of WME `(<emotions> ^|Anger| anger)`
7. Creation of WME `(<emotions> ^|Anger| angry)`
8. Creation of WME `(<emotions> ^|Anger| annoyed)`
9. Creation of WME `(<emotions> ^|Anger| irritated)` (equivalent for other 6 emotions)
10. Creation of WME `(<emotion-responsibility-states> ^|responsible-for-Anger| anger)`
11. Creation of WME `(<emotion-responsibility-states> ^|responsible-for-Anger| angry)`
12. Creation of WME `(<emotion-responsibility-states> ^|responsible-for-Anger| annoyed)`
13. Creation of WME `(<emotion-responsibility-states> ^|responsible-for-Anger| irritated)` (equivalent for other 6 emotions)
14. Creation of WME `(<misc> ^crowd crowd)` (equivalent for other 3 miscelanious words)

Results

All results were met.

C.2 Phase 2

C.2.1 `top-ps*emia*elaborate*lookup-table*static-content*template`

Conditions

1. Rule `top-ps*emia*elaborate*lookup-table*template`.

Test plan

1. Add rule to the system.

Expected results

1. Creation of WME `(<nl-emotion> ^|Anger| |angry|` (equivalent for the other six emotion translations).
2. Creation of WME `(<nl-intensity-lower> ^0 |"a little"|` (equivalent for the other four lower intensity bound translations).
3. Creation of WME `(<nl-intensity-upper> ^0.25 |"a little"|` (equivalent for the other four upper intensity bound translations).
4. Creation of WME `(<nl-polarity> ^negative |not|`.
5. Creation of WME `(<nl-responsible_agent> ^1sldr |Johnson|` (equivalent for the other seven responsible agent pronoun translations).
6. Creation of WME `(<nl-responsible_agent-poss> ^1sldr |"Johnson's"|` (equivalent for the seven responsible agent possesive pronoun translations).
7. Creation of WME `(<nl-state-pre> ^boy-dead |"the boy"|` (equivalent for the other 50 state translations).
8. Creation of WME `(<nl-state-post> ^boy-dead |dead|` (equivalent for the other 50 state translations).
9. Creation of WME `(<nl-status> ^confirmed |being|` (equivalent for the other status translation).

Results

For this rule, only indirect testing was possible as the Soar GUI isn't able to deal with quotes correctly, which results in an inability to show the corresponding WME's. Testing showed the WME's were present internally though, so we can state that all expected results were met.

C.2.2 `top-ps*emia*elaborate*lookup-table*dynamic-content*medic`

Conditions

1. Agent sergeant is used as testing environment.
2. Rule `top-ps*emia*elaborate*lookup-table*template`.

Test plan

1. Add rule to the system.

Expected results

1. Creation of WME `(<nl-responsible_agent> ^medic |Tucci|`.
2. Creation of WME `(<nl-responsible_agent-poss> ^medic |Tucci's|`.

Results

Both expected results were met.

C.2.3 top-ps*emia*elaborate*lookup-table*dynamic-content*mom

Conditions

1. Agent sergeant is used as testing environment.
2. Rule `top-ps*emia*elaborate*lookup-table*template`.

Test plan

1. Add rule to the system.

Expected results

1. Creation of WME `(<nl-responsible_agent> ^mom |The mother|`.
2. Creation of WME `(<nl-responsible_agent-poss> ^mom |the mother's|`.

Results

Both expected results were met.

C.2.4 top-ps*emia*elaborate*lookup-table*dynamic-content*sgt

Conditions

1. Agent sergeant is used as testing environment.
2. Rule `top-ps*emia*elaborate*lookup-table*template`.

Test plan

1. Add rule to the system.

Expected results

1. No creation of WME's.

Results

The expected result was met.

C.2.5 top-ps*emia*elaborate*lookup-table*dynamic-content*self

Conditions

1. Agent sergeant is used as testing environment.
2. Rule `top-ps*emia*elaborate*lookup-table*template`.

Test plan

1. Add rule to the system.

Expected results

1. Creation of WME `(<nl-responsible_agent> ^sgt |I am|`.
2. Creation of WME `(<nl-responsible_agent-poss> ^sgt |my|`.

Results

Both expected results were met.

C.2.6 top-ps*emia*emotion-state*add*causality

Conditions

1. Initialized agent sergeant is used as testing environment; the intensity for hope is 0.57, the intensity for distress is 0.56 and the intensity for the other emotions are about 0. The appraisal with highest intensity concerning the emotion hope appraises the state `aa-secure`.
2. Rule `top-ps*emia*elaborate*emotion-state*max-feeling`

Test plan

1. Add rule to the system.

Expected results

1. Creation of WME `(<max-feeling-emotion-state> ^cause aa-secure)`.
2. Creation of WME `(<max-feeling-emotion-state> ^source <appraisal-object>)`.

Since the construction for the `feeling` emotion state is equal to that of the `max-feeling` emotion states, we can safely assume that if the above tests are executed correctly, the causality is also added correctly to the `feeling` emotion states.

Results

All expected results were met.

C.2.7 top-state*apply*operator*understand-speech*nlu* how-do-you-feel-about-state*one

Conditions

Rule `top-state*elaborate*state*add-lexicon-entries*emotions`

Test plan

1. Add rule to the system.
2. Ask "How do you feel about the crowd?"
3. Ask "How do you feel about the boy?"
4. Ask a few samples at random.

Expected results

After first step of test plan: rule should not fire. After second step of test plan:

1. Creation of WME `(^interpretation mood question)`
2. Creation of WME `(^interpretation emotion-semantics <emotion-semantics>)`
3. Creation of WME `(<emotion-semantics> ^type question)`
4. Creation of WME `(<emotion-semantics> ^q-slot value)`
5. Creation of WME `(<emotion-semantics> ^priority 1)`
6. Creation of WME `(<emotion-semantics> ^prop <sem>)`
7. Creation of WME `(<sem> ^attribute feeling-about-state)`
8. Creation of WME `(<sem> ^type state)`
9. Creation of WME `(<sem> ^cause maintain-goodwill)`

After third step of test plan: idem as step two, but mapping to all states concerning the boy, choosing one of them for the `cause`:

1. `boy-dead`
2. `boy-healthy`
3. `minor-injuries`
4. `serious-injuries`
5. `critical-injuries`
6. `know-boy-health`
7. `at-boy-aa`
8. `at-boy-hospital`

After fourth step of test plan: idem as step two, but mapping `cause` to the appropiate state.

The fourth step of the test plan is a compromise between testing all states and being time efficient. Together with the testing of similar rules for two and three keywords for states, and the lexicon testing, this should provide enough security about the correctness of the code.

Results

The third part of the test plan went not according to plan, as the states `minor-injuries`, `serious-injuries` and `critical-injuries` were pointed to twice and the state `driver-minor-inj` also mapped to "boy". This is due to how te original lexicon was set up, namely diveded into various domains. As several domains must be used by the emotion code in order to find the necessary synonyms, the code scans all the domains in the lexicon. By adding a specific `states` domain, this too is included in the search and so "boy" can lead to "minor-injuries" and from there to either to the state `minor-injuries` or `driver-minor-inj`. To avoid these situations the Soar code for this rule (and the similar rules for two and three keywords) has been changed, so it will exclude the `states` domain. Also, the `emotion-responsibility-states` entry is excluded as this led to conflicts when using emotions for state detection.

Furthermore, there were some minor issues with both the existing as the emotion lexicon, like spelling errors and missing synonyms. These were fixed.

C.2.8 `top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*two`

Conditions

Rule `top-state*elaborate*state*add-lexicon-entries*emotions`

Test plan

1. Add rule to the system.
2. Ask "How do you feel about the boy being injured?".
3. Ask "How do you feel about the driver being healthy?".
4. Ask "How do you feel about the route being secured?".
5. Ask a few samples at random.

Expected results

After first step of test plan: rule should not fire. After second step of test plan:

1. Creation of WME `(^interpretation mood question)`
2. Creation of WME `(^interpretation emotion-semantics <emotion-semantics>)`
3. Creation of WME `(<emotion-semantics> ^type question)`
4. Creation of WME `(<emotion-semantics> ^q-slot value)`
5. Creation of WME `(<emotion-semantics> ^priority 2)`
6. Creation of WME `(<emotion-semantics> ^prop <sem>)`
7. Creation of WME `(<sem> ^attribute feeling-about-state)`
8. Creation of WME `(<sem> ^type state)`

9. Creation of WME (<sem> ^cause [min/ser/crit]-injuries)

After third step of test plan: idem as step two, but mapping cause to state driver-healthys.

After fourth step of test plan: idem as step two, but mapping cause to state route-secure.

After fith step of test plan: idem as step two, but mapping cause to the appropiate state.

As the principal for all other states is equivalent and the lexicon rule has already been tested, we can savely assume that if the above tests are executed correctly, this holds true for all states.

Results

There were some minor issues with both the existing as the emotion lexicon, like spelling errors and missing synonyms. These were fixed.

C.2.9 top-state*apply*operator*understand-speech*nlu* how-do-you-feel-about-state*three

Conditions

Rule top-state*elaborate*state*add-lexicon-entries*emotions

Test plan

1. Add rule to the system.
2. Ask "How do you feel about the boy having serious injuries?".
3. Ask "How do you feel about the minor injuries of the boy?".
4. Ask "How do you feel about the driver having minor injuries?".
5. Ask a few samples at random.

Expected results

After first step of test plan: rule should not fire. After second step of test plan:

1. Creation of WME (^interpretation mood question)
2. Creation of WME (^interpretation emotion-semantics <emotion-semantics>)
3. Creation of WME (<emotion-semantics> ^type question)
4. Creation of WME (<emotion-semantics> ^q-slot value)
5. Creation of WME (<emotion-semantics> ^priority 3)
6. Creation of WME (<emotion-semantics> ^prop <sem>)
7. Creation of WME (<sem> ^attribute feeling-about-state)
8. Creation of WME (<sem> ^type state)

9. Creation of WME `(<sem> ^cause serious-injuries)`

After third step of test plan: idem as step two, but mapping `cause` to state `minor-injuries`.

After fourth step of test plan: idem as step two, but mapping `cause` to state `driver-minor-inj`.

After fifth step of test plan: idem as step two, but mapping `cause` to the appropiate state.

As the principal for all other states is equivalent and the lexicon rule has already been tested, we can savely assume that if the above tests are executed correctly, this holds true for all states.

Results

There were some minor issues with both the existing as the emotion lexicon, like spelling errors and missing synonyms. These were fixed.

C.2.10 `top-state*apply*operator*understand-speech*nlu*do-you-feel-emotion`

Conditions

Rule `top-state*elaborate*state*add-lexicon-entries*emotions`

Test plan

1. Add rule to the system.
2. Ask "Why do you feel guilty?"
3. Ask "Who's responsible for making you feel angry?"
4. Ask "Are you angry?".
5. Ask "Do you feel anger?".
6. Ask "You seem worried".
7. Ask "You look guilty".
8. Ask a few samples at random.

Expected results

After the first, second and third step of test plan: rule should not fire. After fourth step of test plan:

1. Creation of WME `(^interpretation mood question)`
2. Creation of WME `(^interpretation emotion-semantics <emotion-semantics>)`
3. Creation of WME `(<emotion-semantics> ^type question)`
4. Creation of WME `(<emotion-semantics> ^q-slot polarity)`

5. Creation of WME (<emotion-semantics> ^prop <sem>)
6. Creation of WME (<sem> ^attribute feeling)
7. Creation of WME (<sem> ^type state)
8. Creation of WME (<sem> ^value |Anger|)

After fifth step of test plan: idem as step four
After sixth step of test plan: idem as step four, but mapping value to state |Distress|.
After seventh step of test plan: idem as step four, but mapping value to state |Guilt|.
After eighth step of test plan: idem as step four, but mapping value to the appropiate emotion.
As the principal for all emotions is equivalent and the lexicon rule has already been tested, we can savely assume that if the above tests are executed correctly, this holds true for all emotions.

Results

As Soar is case sensive, the question "Why do you feel guilty" did trigger this rule, because it didn't contain the word "why", but "Why". It will be stated that alle sentences should be typed in lower case characters. All other results were met.

C.2.11 top-state*apply*operator*understand-speech*nlu* why-do-you-feel-emotion

Conditions

Rule top-state*elaborate*state*add-lexicon-entries*emotions

Test plan

1. Add rule to the system.
2. Ask "Why do you feel guilty?"
3. Ask "Why are you worried?"
4. Ask a few samples at random.

Expected results

After the first step of test plan: rule should not fire. After second step of test plan:

1. Creation of WME (^interpretation mood question)
2. Creation of WME (^interpretation emotion-semantics <emotion-semantics>)
3. Creation of WME (<emotion-semantics> ^type question)
4. Creation of WME (<emotion-semantics> ^q-slot cause)

5. Creation of WME (<emotion-semantics> ^prop <sem>)
6. Creation of WME (<sem> ^attribute feeling)
7. Creation of WME (<sem> ^type state)
8. Creation of WME (<sem> ^value |Guilt|)

After third step of test plan: idem as step four, but mapping value to state |Distress|.

Results

The question "Why do you feel guilty?" let to the firing of both ...*nlu* why-do-you-feel-emotion and ...*nlu*responsible-for-emotion*guil, because of overlapping keywords. It showed that the latter scanned for certain keywords which were'nt necessary. This was fixed. All other results were met.

C.2.12 top-state*apply*operator*understand-speech*nlu* responsible-for-emotion proc stateResponsible

As the rule top-state*apply*operator*understand-speech*nlu*responsible-for-emotion is used together with proc stateResponsible, we test them in one test.

Conditions

Rule top-state*elaborate*state*add-lexicon-entries*emotions

Test plan

1. Add rule to the system.
2. Ask "Whos responsible for making you feel hopeful?"
3. Ask a few samples at random.

Expected results

After the first step of test plan: rule should not fire. After second step of test plan:

1. Creation of WME (^interpretation mood question)
2. Creation of WME (^interpretation emotion-semantics <emotion-semantics>)
3. Creation of WME (<emotion-semantics> ^type question)
4. Creation of WME (<emotion-semantics> ^q-slot value)
5. Creation of WME (<emotion-semantics> ^prop <sem>)
6. Creation of WME (<sem> ^attribute responsible-for-Hope)
7. Creation of WME (<sem> ^type state)

After third step of test plan: idem as step four, but mapping `attribute` to state `|Distress|`.

As the design is similar for the rule `...nlu*responsible-for-emotion*guilt`, we can savely assume that if the above tests are executed correctly and if during the fourth step these rules are coverd, this holds true for both rules.

Results

All expected results were met.

C.3 Phase 3

C.3.1 `top-ps*emia*emotion-state*add*causality*to*negation`

Conditions

1. Initialized agent sergeant is used as testing environment; the intensity for hope is 0.57, the intensity for distress is 0.56 and the intensity for the other emotions are about 0. The appraisal with highest intensity concerning the emotion hope appraises the state `at-boy-hospital`.
2. Rule `top-ps*emia*elaborate*emotion-state*max-feeling`
3. Rule `top-ps*emia*emotion-state*add*causality`

Test plan

1. Add rule to the system.

Expected results

1. Creation of WME `(<max-feeling-emotion-state-negated> ^cause at-boy-hospital)`.
2. Creation of WME `(<max-feeling-emotion-state-negated> ^source <appraisal-object>)`.

Since the construction for the `feeling` emotion state is equal to that of the `max-feeling` emotion states, we can safely assume that if the above tests are executed correctly, the causality is also added correctly to the `feeling` emotion states.

Results

All expected results were met.

C.3.2 `top-ps*emia*elaborate*emotion-state*responsible-for-[emotion]`

Conditions

1. Initialized agent sergeant is used as testing environment; the intensity for hope is 0.57, the intensity for distress is 0.56 and the intensity for the other emotions are about 0. The appraisal with highest intensity concerning the emotion hope appraises the state `at-boy-hospital`. The responsible agent for this state is the medevac.

2. Rule `top-ps*emia*elaborate*emotion-state*feeling`
3. Rule `top-ps*emia*emotion-state*add*causality`

Test plan

1. Add rule to the system.

Expected results

1. Creation of WME `(<current-state> ^<new1> <newname1>)`.
2. Creation of WME `(<current-state> ^<new2> <newname2>)`.
3. Creation of WME `(<newname1> ^attribute responsible-for-Anger)`.
4. Creation of WME `(<newname1> ^belief true)`.
5. Creation of WME `(<newname1> ^cause praiseworthy)`.
6. Creation of WME `(<newname1> ^polarity positive)`.
7. Creation of WME `(<newname1> ^value medevac)`.
8. Creation of trivial WME's concerning object `<newname1>`.
9. Creation of object `<newname2>`, the inverse of `<newname1>`.
10. Creation of objects, equal to `<newname1>` and their inverses, but with `^attribute responsible-for-Anxiety`, `^attribute responsible-for-Distress`, `^attribute responsible-for-Fear`, `^attribute responsible-for-Hope`, `^attribute responsible-for-Guilt`, `^attribute responsible-for-Joy`, *only* if a causality exists.

Results

All expected results were met.

C.3.3 `top-state*apply*operator*understand-speech*emotion-semantics-priority`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*two` or top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*three

Test plan

1. Add rule to the system.
2. Ask "How do you feel about the boy?".
3. Ask "How do you feel about the boy being injured?".
4. Ask "How do you feel about the driver having minor injuries?"

Expected results

After the first and second step of the test plan, the rule should not fire.

After the third step of the test plan, there are only `emotion-semantics` attributes with priority 2.

Afther the fourth step of the test plan, there are only `emotion-semantics` attributes with priority 3.

Results

All expected results were met.

C.3.4 `top-state*apply*operator*understand-speech* emotion-semantics-selection*same`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*two` or top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*three

Test plan

1. Add rule to the system.
2. Ask "How do you feel about the boy?".
3. Ask "How do you feel about the boy being injured?".
4. Ask "How do you feel about the driver having minor injuries?"

Expected results

After the first step of the test plan, the rule should not fire.

After the second step of the test plan, there's eventually only one `emotion-semantics` with has priority 1.

After the third step of the test plan, there's eventually only one `emotion-semantics` with has priority 2.

After the fourth step of the test plan, there's eventually only one `emotion-semantics` with has priority 3.

Results

All expected results were met.

C.3.5 `top-state*apply*operator*understand-speech* emotion-semantics-selection*cause-vs-rebel-without-a-cause`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*two` or top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*three

Test plan

1. Add rule to the system.
2. Ask "How do you feel about the boy?".
3. Ask "How do you feel about the boy being injured?".
4. Ask "How do you feel about the driver having minor injuries?"

Expected results

After the first step of the test plan, the rule should not fire.

The other steps should result in all `emotion-states` which have cause; all other states should be discarded.

Results

All expected results were met.

C.3.6 `top-state*apply*operator*understand-speech*emotion-semantics-selection*intensity-vs-no-intensity`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*two` or top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*three

Test plan

1. Add rule to the system.
2. Ask "How do you feel about the boy?".
3. Ask "How do you feel about the boy being injured?".
4. Ask "How do you feel about the driver having minor injuries?"

Expected results

After the first step of the test plan, the rule should not fire.

The other steps should result in all `emotion-states` which have an intensity; all other states should be discarded.

Results

All expected results were met.

C.3.7 top-state*apply*operator*understand-speech* emotion-semantics-preference*sem

Conditions

1. Rule top-state*apply*operator*understand-speech*nlu* how-do-you-feel-about-state*two or top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*three

Test plan

1. Add rule to the system.
2. Ask "How do you feel about the boy?".
3. Ask "How do you feel about the boy being injured?".
4. Ask "How do you feel about the driver having minor injuries?"

Expected results

After the first step of the test plan, the rule should not fire.

The other steps should result in one semantics under interpretation, with the same attributes and values as emotion-semantics

This rule is simular to ...*emotion-semantics-preference*semantics, so if these test are executed correctly, we can savely assume both rules arre correct.

Results

All expected results were met.

C.4 Phase 4

This section will exclusively test the code which is responsible for selecting and outputting the agents' utterances. As an utterance is being produced by the combination of both a Soar rule and a TCL template, these will be tested together.

The first two tests will test all mechanisms of the template. As all the templates are constructed according to the same design, subsequent template features will only be tested randomly for the sake of time.

All the rules are dependend on the rules below. Additional depencies will be depicted in the according subsection.

The Initiative value of the agent is set to zero.

1. Rule top-state*apply*operator*understand-speech*nlu*how-do-you-feel or top-state*apply*operator*understand-speech*nlu*whats-wrong-with-you
2. Rule top-state*apply*operator*understand-speech*emotion-semantics-priority
3. Rule top-state*apply*operator*understand-speech*emotion-semantics-selection
4. Rule top-state*apply*operator*understand-speech*emotion-semantics-preference*sem

5. Rule `top-state*apply*operator*understand-speech*emotion-semantics-preference*semantics`
6. Rule `top-state*apply*operator*understand-speech*emotion-semantics-interpretation-semantics`
7. Rule `top-ps*emia*elaborate*lookup-table*static-content*template`
8. Rule `top-ps*emia*elaborate*lookup-table*dynamic-content*medic`
9. Rule `top-ps*emia*elaborate*lookup-table*dynamic-content*mom`
10. Rule `top-ps*emia*elaborate*lookup-table*dynamic-content*sgt`
11. Rule `top-ps*emia*elaborate*lookup-table*dynamic-content*self`

C.4.1 `top-state*apply*operator*output-speech*answer-how-do-you-feel*cause proc emotionTypeFeelCause`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*how-do-you-feel`
2. Rule `top-ps*emia*elaborate*emotion-state*max-feeling`
3. Rule `top-ps*emia*emotion-state*add*source`
4. Rule `top-ps*emia*emotion-state*add*cause`

Test plan

1. Add rules to the system.
2. Ask "How do you feel?".
3. Set Hope < 0.2, ask "How do you feel?"
4. Set Distress > 0.75, ask "How do you feel?"
5. Set Distress $> 0.25 < 0.5$, ask "How do you feel?"
6. Set Distress $> 0.2 < 0.25$, ask "How do you feel?"
7. Set Terseness> 0.2, ask "How do you feel?"
8. Set Terseness > 0.6, ask "How do you feel?"
9. Set Distress < 0.2, ask "How do you feel?"

Expected results

After the first step of the test plan, the rule should not fire.

After the second step of the plan, the agent should utter "I'm hopeful because the boy will probably be treated at the hospital".

After the third step of the plan, the agent should utter "I'm distressed because the boy is critically injured".

After the fourth step of the plan, the agent should utter "I'm very distressed because the boy is critically injured".

After the fifth step of the plan, the agent should utter "I'm pretty distressed because the boy is critically injured".

After the sixth step of the plan, the agent should utter "I'm a bit distressed because the boy is critically injured".

After the seventh step of the plan, the agent should utter "I'm a bit distressed".

After the eigth step of the plan, the agent should utter "I'm distressed".

After the ninth step of the plan, the agent should utter "I'm fine sir".

Results

All expected results were met, although sometimes the agent answers twice, either directly after each other, or seperated by the phrase "Well". This will be reported as a bug.

C.4.2 `top-state*apply*operator*output-speech* answer-how-do-you-feel*no-cause proc emotionTypeFeelNoCause`

Conditions

1. Hope set to 0
2. Distress set to 0
3. Guilt set $> 0 < 0.2$
4. Rule `top-state*apply*operator*understand-speech*nlu*how-do-you-feel`
5. Rule `top-ps*emia*elaborate*emotion-state*max-feeling`

Test plan

1. Add rules to the system.
2. Ask "How do you feel?".
3. Set Guilt $> 0.2 < 0.25$, ask "How do you feel?"
4. Set Guilt $> 0.25 < 0.5$, ask "How do you feel?"
5. Set Guilt $> 0.5 < 0.75$, ask "How do you feel?"
6. Set Guilt > 0.75, ask "How do you feel?"
7. Set Terseness > 0.6, ask "How do you feel?"

Expected results

After the first step of the test plan, the rule should not fire.

After the second step of the plan, the agent should utter "I'm fine sir".

After the third step of the plan, the agent should utter "I'm a little guilty".

After the fourth step of the plan, the agent should utter "I'm pretty guilty".

After the fifth step of the plan, the agent should utter "I'm guilty".

After the sixth step of the plan, the agent should utter "I'm very guilty".

After the seventh step of the plan, the agent should utter "I'm guilty".

Results

All expected results were met. It is noted that the template doesnt produce good natural language for "guilt". This is something to be looked at.

C.4.3 `apply*output-speech*answer-do-you-feel-emotion*cause proc emotionTypeEmotionCause`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*how-do-you-feel`
2. Rule `top-ps*emia*elaborate*emotion-state*feeling*true`
3. Rule `top-ps*emia*elaborate*emotion-state*feeling*false`
4. Rule `top-ps*emia*emotion-state*add*source`
5. Rule `top-ps*emia*emotion-state*add*cause`

Test plan

1. Add rules to the system.
2. Ask "Do you feel hopeful?".
3. Ask "Are you distressed?".
4. Set Terseness set to$>$ 0.25, ask "Do you feel concerned?"
5. Set Distress set to $> 0.1 < 0.2$, ask "Are you distressed?"
6. Ask similar question with emotion and terseness values changed randomly

Expected results

After the first step of the test plan, the rule should not fire.

After the second step of the plan, the agent should utter "I do sir because the boy will probably be treated at the hospital".

After the third step of the plan, the agent should utter "I do sir because the boy is critically injured".

After the fourth step of the plan, the agent should utter "I sure do sir".

After the fifth step of the plan, the agent should utter "No sir".

After the sixth step of the plan, the agent should utter according to the question

Results

All expected results were met. It is noted that the template doesnt differentiate between the verbs "are" and "feel". This is something to be looked at.

C.4.4 `apply*output-speech*answer-do-you-feel-emotion*no-cause proc emotionTypeEmotionNoCause`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*how-do-you-feel`
2. Rule `top-ps*emia*elaborate*emotion-state*feeling*true`
3. Rule `top-ps*emia*elaborate*emotion-state*feeling*false`

Test plan

1. Add rules to the system.
2. Ask "Are you happy?"
3. Set Anxiety $> 0.1 < 0.2$, ask "Do you feel anxious?"
4. Ask similar question with emotion and terseness values changed randomly

Expected results

After the first step of the test plan, the rule should not fire.
After the second step of the plan, the agent should utter "Not at all sir".
After the third step of the plan, the agent should utter "No sir".
After the fourth step of the plan, the agent should utter according to the question

Results

All expected results were met.

C.4.5 `top-state*apply*operator*output-speech* answer-emotion-about-state*cause proc emotionTypeState`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu* how-do-you-feel-about-state*one` or `top-state*apply*operator*understand-speech*nlu* how-do-you-feel-about-state*two` or `top-state*apply*operator*understand-speech*nlu* how-do-you-feel-about-state*three`
2. Rule `top-ps*emia*elaborate*emotion-state*feeling-about-state`

Test plan

1. Add rules to the system.
2. Ask "How do you feel about the boy being critical injured?"
3. Set Hope to > 0.75, ask "How do you feel about the boy going to the hospital?"
4. Ask similar question with emotion and terseness values changed randomly

Expected results

After the first step of the test plan, the rule should not fire.

After the second step of the plan, the agent should utter "I'm distressed about it sir".

After the third step of the plan, the agent should utter "I'm very hopeful about it sir".

After the fourth step of the plan, the agent should utter according to the question

Results

It came to light that the Terseness attribute isn't always present. This prevents our rules from being fired. It will reported as a bug to the MRE team. Next to that, the expected result only turned up now and then. This will be treated as a bug.

C.4.6 `top-state*apply*operator*output-speech*answer-emotion-about-state*no-cause`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*one` or `top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*two` or `top-state*apply*operator*understand-speech*nlu*how-do-you-feel-about-state*three`
2. Rule `top-ps*emia*elaborate*emotion-state*feeling-about-state`

Test plan

1. Add rules to the system.
2. Ask "How do you feel about the route being secured?"
3. Ask "How do you feel about the crowd?"

Expected results

After the first step of the test plan, the rule should not fire.

After the second and third step of the plan, the agent should utter "That's not an issue right now sir".

Results

All the expected results were met.

C.4.7 `top-state*apply*operator*output-speech*answer-calm-down proc emotionCalm`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*calm-down` or `top-state*apply*operator*understand-speech*nlu*relax`
2. Rule `top-ps*emia*elaborate*emotion-state*feeling-about-state`

Test plan

1. Add rules to the system.
2. Say "Calm down, you maniac!"
3. Set Hope and Distress to < 0.2, "You'd better relax..."
4. Set Fear to > 0.2, say "Calm down, soldier, get a hold of yourself! Wasn't I always there for you in times of need? Haven't I always covered your back? Looking after you in even the most dangerous of situations? Pull yourself together and give those mother****ers hell!!"

Expected results

After the first step of the test plan, the rule should not fire.
After the second step of the plan, the agent should utter "Will do sir".
After the third step of the plan, the agent should utter "I am calm, sir".
After the fourth step of the plan, the agent should utter "You're right, sir! We can make it togheter! You've always stood by my side, and I respect you for that; both as my superior, and... as my friend,", after which the lieutenant and the sergeant embrace each other, assemble their men, and full of spirit face the enemy on the battlefield where they will ultimately die a heroes death.

Results

The first step showed that the Soar rule matches twice, which is not necessary. The predicate `belief true` was therefore added, as was already the case in the other Soar rules. All expected results showed, though, except for the fourth one, which was actually a little joke.

C.4.8 `apply*output-speech*answer-why-do-you-feel-emotion*cause proc emotionWhyCause`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*why-do-you-feel-emotion` or `top-state*apply*operator*understand-speech*nlu*whats-causing-you-to-feel-emotion`
2. Rule `top-ps*emia*elaborate*emotion-state*feeling*true`

3. Rule `top-ps*emia*emotion-state*add*source`

4. Rule `top-ps*emia*emotion-state*add*cause`

Test plan

1. Add rules to the system.

2. Ask "Why do you feel hopeful?"

3. Set Devensiveness > 0.75, Ask "Why do you feel hopeful?"

4. Ask appropiate question, changing emotions and Terseness randomly

Expected results

After the first step of the test plan, the rule should not fire.

After the second step of the plan, the agent should utter "Because the boy will probably be treated at the hospital sir".

After the third step of the plan, the agent should utter "That's personal sir".

After the fourth step of the plan, the agent should react accordingly.

Results

All expected results were met. As it is hard to get the agent in an state where he really blames someone, the associated output was tested using slightly altered versions of the code; all expected results were met.

C.4.9 `apply*output-speech*answer-why-do-you-feel-emotion* no-responsible-agent proc emotionWhyNoResponsibleAgent`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu* why-do-you-feel-emotion` or `top-state*apply*operator*understand-speech*nlu* whats-causing-you-to-feel-emotion`

2. Rule `top-ps*emia*elaborate*emotion-state*feeling*true`

3. Rule `top-ps*emia*emotion-state*add*cause`

Test plan

1. Add rules to the system.

2. Ask "Why are you distressed?"

3. SeEt Devensiveness > 0.75, Ask "Why are you distressed?"

4. Set Distress < 0.2, ask "Why do you feel distressed?"

Expected results

After the first step of the test plan, the rule should not fire.

After the second step of the plan, the agent should utter "Because the boy is critically injured sir".

After the third step of the plan, the agent should utter "That's personal sir".

After the fourth step of the plan, the agent should utter "I'm not distressed sir".

Results

All expected results were met.

C.4.10 `apply*output-speech*answer-why-do-you-feel-emotion*no-cause proc emotionWhyNoCause`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu* why-do-you-feel-emotion` or `top-state*apply*operator*understand-speech*nlu* whats-causing-you-to-feel-emotion`
2. Rule `top-ps*emia*elaborate*emotion-state*feeling*true`

Test plan

1. Add rules to the system.
2. Ask "Why are you anxious?"
3. Set Anxiety > 0.1, Ask "Why do you feel anxious?"

Expected results

After the first step of the test plan, the rule should not fire.

After the second step of the plan, the agent should utter "I'm not anxious sir".

After the third step of the plan, the agent should utter "I don't know sir".

Results

Expected results sometimes showed up, but there was a lot of interference with the original code which handles the why-question. This will be discussed with the MRE team.

C.4.11 `apply*output-speech*answer-whos-responsible proc emotionResponsibility`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu*responsible-for-emotion` or `top-state*apply*operator*understand-speech*nlu*responsible-for-emotion*guilt`
2. Rule `top-ps*emia*elaborate*emotion-state*responsible-for-[emotion]`

Test plan

1. Add rules to the system.
2. Ask "Who's responsible for making you feel hopeful?"
3. Set Hope < 0.1, Ask "Who's responsible for making you feel hopeful?"

Expected results

After the first step of the test plan, the rule should not fire.
After the second step of the plan, the agent should utter "The medevac sir".
After the third step of the plan, the agent should utter "I'm not hopeful sir".

Results

Some minor issues concerning the template filling showed up; these were fixed. As it is hard to get the agent in an state where he really blames someone, the associated output was tested using slightly altered versions of the code; all expected results were met.

C.4.12 `apply*output-speech*answer-whos-responsible* no-responsible-agent proc emotionNoResponsibility`

Conditions

1. Rule `top-state*apply*operator*understand-speech*nlu* responsible-for-emotion` or `top-state*apply*operator*understand-speech*nlu* responsible-for-emotion*guilt`
2. Rule `top-ps*emia*elaborate*emotion-state*responsible-for-[emotion]`

Test plan

1. Add rules to the system.
2. Ask "Who's responsible for making you feel guilty?"
3. Set Guilty > 0.2, ask "Who's responsible for making you feel guilty?"

Expected results

After the first step of the test plan, the rule should not fire.
After the second step of the plan, the agent should utter "I'm not guilty sir".
After the third step of the plan, the agent should utter "I don't know sir".

Results

Some minor issues concerning the template filling showed up; these were fixed.

C.5 Missing tests

e rules `top-ps*emia*elaborate*emotion-state*feeling-about-state` and `top-ps*emia*emotion-state*feeling-about-state*add*sim-object` have not been unit tested, because of the difficulty to create an environment with the proper left hand side. The first rule needs an operator, but it is not possible in Soar to hand select the operator at any given point in time. The second rule has the first as a requirement, so without it, it cannot be tested.

Bibliography

[Cas00] Justine Cassell: Nudge Nudge Wink Wink: Elements of Face-to-Face Conversation for Embodied Conversational Agents, Embodied Conversational Agents, chapter 1, edited by Justine Cassell, et al. (2000)

[Cho99] S. Chopra-Khullar, N.I. Badler: Where to Look? Automating Attending Behaviors of Virtual Human Characters, proc. 3rd Int'l Conf. Autonomous Agents, ACM Press, New York, pp. 16-23 (1999)

[Gra04] Jonathan Gratch, Stacy Marsella: A Domain-independent Framework for Modeling Emotional Appraisal and Coping

[Gra02] Jonathan Gratch, Jeff Richel, Elisabeth André, Justine Cassell, Eric Petajan, Norman Badler: Creating Interactive Virtual Humans: Some Assembly Required

[Hil99] Randall Hill: Modeling Perceptual Attention in Virtual Humans, Proc 8th Conf. Computer Generated Forces and Behavioral Representation, SISO, Orlando, Fla., pp. 563-573 (1999)

[Hil00] Randall Hill: Perceptual Attention in Virtual Humans: Toward Realistic and Believable Gaze Behaviors, Proc. AAAI Fall Symp. Simulating Human Agents, AAAI Press, Menlo Park, Calif., pp. 46-52 (2000)

[ICT03] Mission Rehearsel Exercise - Institute for Creative Technologies: `http://www.ict.usc.edu/disp.php?bd=proj_mre` (October 22, 2003)

[Jur00] D. Jurafsky, James H. Martin: Speech and language processing (2000) - Prentice Hall, ISBN 0-13-096069-6

[Lai99] John E. Laird, Clare Bates Congdon, Karen J. Coulter: The Soar User Manual (June 23, 1999)

[Mar??] Stacy Marsella, Jonathan Gratch, Jeff Rickel: Expressive Behaviors for Virtual Worlds

[Ric00] Jeff Rickel, W. Lewis Johnson: Task-Oriented Collaboration with Embodied Agents in Virtual Worlds, Embodied Conversational Agents, chapter 4, Edited by Justine Cassell, et al. (2000)

[Ric02] Jeff Rickel, Stacy Marsella, Jonathan Gratch, Randall Hill, David Traum, William Swartout: Toward a New Generation of Virtual Humans for Interactive Experiences (2002)

[Tra??] David R. Traum, Staffan Larsson: The information state approach to dialogue management

[Tra02] David Traum, Jeff Rickel: Emodied Agents for Multi-party Dialogue in Immersive Virtual Worlds (2002)

[Tra03] David Traum, Jeff Rickel, Jonathan Gratch, Stacy Marsella: Negotiation over Tasks in Hybrid Human-Agent Teams for Simulation-Based Training (July 18, 2003)

[Tra03b] David Traum, Michael Fleischman, Eduard Hovy: NL Generation for Virtual Humans in a Complex Social Environment (2003)

[Tra03c] David Traum: Semantics and Pragmatics of Questions and Answers for Dialogue Agents (2003)

Index