Multi Agenten Systeme

VU SS 00, TU Wien

Teil 1 (Kapitel 1–4) basiert auf

Multi-Agent Systems (Gerhard Weiss), MIT Press, June 1999.

Es werden allgemeine Techniken und Methoden dargestellt (BDI-, Layered-, Logic based Architekturen, Decision Making, Kommunikation/Interaktion, Kontrakt Netze, Coalition Formation).

Teil 2 (Kapitel 5–9) basiert auf

Heterogenous Active Agents (Subrahmanian, Bonatti, Dix, Eiter, Kraus, Özcan and Ross), MIT Press, May 2000.

Hier wird ein spezifischer Ansatz vorgestellt, der formale Methoden aus dem logischen Programmieren benutzt, aber nicht auf PROLOG aufsetzt (Code Call Mechanismus, Aktionen, Agenten Zyklus, Status Menge, Semantiken, Erweiterungen um Beliefs, Implementierbarkeit).

Übersicht

9. Meta Agent Programs

Overview

Timetable:

- Chapter 9 needs 1 lecture, but lots of things need to be done quickly.

9 Meta Agent Programs

9.1 Extending CFIT: CFIT*

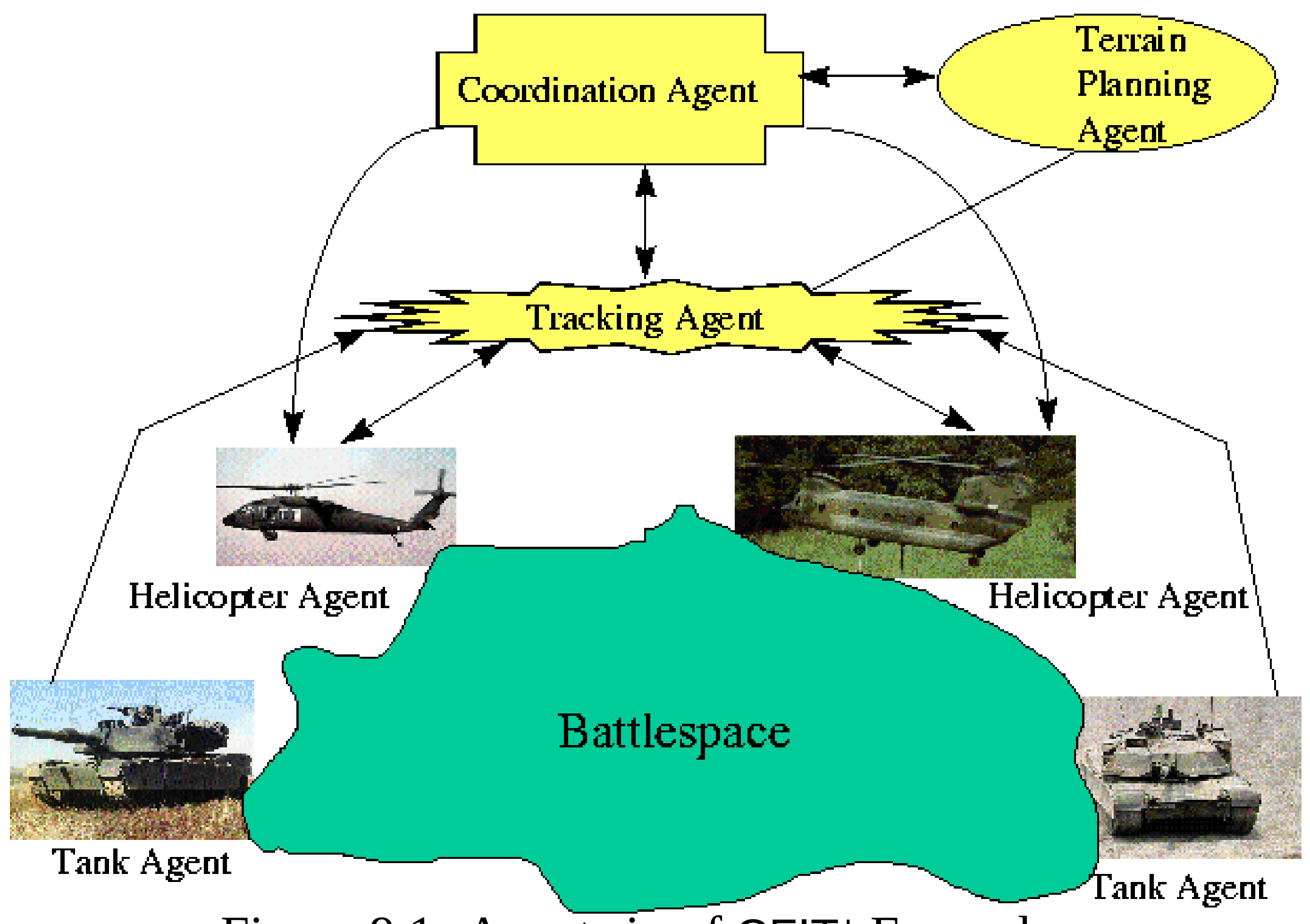

Figure 9.1: Agents in of CFIT* Example

A set of enemy vehicle agents: These agents (mostly tanks) move across free terrain, and their movements are determined by a program that the other agents listed below do not have access to (though they may have beliefs about this program).

A terrain route planning agent `terrain`, (see Table 5.2). Here we extend the `terrain` agent so that it also provides a flight path computation service for helicopters, through which it plans a flight, given an origin, a destination, and a set of constraints specifying the height at which the helicopters wish to fly.

A tracking agent, which takes as input, a *DTED* (Digital Terrain Elevation Data) map, an `id` assigned to an enemy agent, and a time point. It produces as output, the location of the enemy agent at the given point in time (if known) as well as its best guess of what kind of enemy the agent is.

A coordination agent, that keeps track of current friendly assets. This agent receives input and ships requests to the other agents with a view to determining exactly what target(s) the enemy columns may be attempting to strike, as well as determining how to nullify the oncoming convoy.

A set of helicopter agents, that may receive instructions from the coordination agent about when and where to attack the enemy vehicles. When such instructions are received, the helicopter agents contact the terrain route planning agent, and request a flight path. Such a flight path uses terrain elevation information (to ensure that the helicopter does not fly into the side of a mountain).

- *Beliefs about the type of enemy vehicle.* Each enemy vehicle has an associated type—for example, one vehicle may be a T-80 tank, the other may be a T-72 tank. However, the coordination agent may not precisely know the type of a given enemy vehicle, because of inaccurate and/or uncertain identification made by the sensing agent.

  At any point in time, it holds some beliefs about the identity of enemy vehicle.

- *Beliefs about intentions of enemy vehicle.* The coordination agent must try to guess what the enemy's target is.

- *Changing beliefs with time.* As the enemy agent continues along its route, the **coordination** agent may be forced to revise its beliefs, as it becomes apparent that the actual route being taken by the enemy vehicle is inconsistent with the expected route. Furthermore, as time proceeds, sensing data provided by the **tracking** agent may cause the **coordination** agent to revise its beliefs about the enemy vehicle type.
- *Beliefs about the enemy agent's reasoning.* The **coordination** agent may also hold some beliefs about the enemy agents' reasoning capabilities (see the *Belief-Semantics Table* in Definition 9.4 on page 421). For instance, with a relatively unsophisticated and disorganized enemy whose command and control facilities have been destroyed, it may believe that the enemy does not know what moves friendly forces are making.

9.2 Belief Language and Data Structures

- When an agent $\mathfrak{a}$ reasons about another agent $\mathfrak{b}$, it must have some beliefs about $\mathfrak{b}$'s underlying action base (*what actions can* $\mathfrak{b}$ *take?*), $\mathfrak{b}$'s action program (*how will* $\mathfrak{b}$ *reason?*) etc.
- Let us denote this by the belief atom

$$\mathcal{B}_{\mathfrak{a}}(\mathfrak{b}, \chi)$$

which represents one of the beliefs of agent $\mathfrak{a}$ about what holds in the state of agent $\mathfrak{b}$.

- In that case, agent $\mathfrak{a}$ must also have **background information**: beliefs about agent $\mathfrak{b}$'s software package $\mathcal{S}^{\mathfrak{b}}$: the code call condition χ has to be contained in $\mathcal{S}^{\mathfrak{b}}$. We will collect all the beliefs that an agent $\mathfrak{a}$ has about another agent $\mathfrak{b}$ in a set $\Gamma^{\mathfrak{a}}(\mathfrak{b})$ (see Definition 9.10 on page 448).

Definition 9.1 (Belief Atom/Literal, $\mathcal{B}\mathbf{At}_1(\mathfrak{a},\mathfrak{b})$, $\mathcal{B}\mathbf{Lit}_1(\mathfrak{a},\mathfrak{A})$)
Let $\mathfrak{a},\mathfrak{b}$ be agents in $\mathfrak{A}$. Then we define the set $\mathcal{B}\mathbf{At}_1(\mathfrak{a},\mathfrak{b})$ of $\mathfrak{a}$-belief atoms about $\mathfrak{b}$ of level 1 as follows:

1. *If χ is a compatible code call condition of $\mathfrak{a}$ with respect to $\mathfrak{b}$, then $\mathcal{B}_{\mathfrak{a}}(\mathfrak{b},\chi)$ is a* belief atom.
2. *For $Op \in \{\mathbf{O},\mathbf{W},\mathbf{P},\mathbf{F},\mathbf{Do}\}$: if $\alpha(\vec{t})$ is a compatible action atom of agent $\mathfrak{a}$ with respect to $\mathfrak{b}$, then $\mathcal{B}_{\mathfrak{a}}(\mathfrak{b},Op\,\alpha(\vec{t}))$ is a* belief atom.

If $\mathcal{B}_{\mathfrak{a}}(\mathfrak{b},\chi)$ is a belief atom, then $\mathcal{B}_{\mathfrak{a}}(\mathfrak{b},\chi)$ and $\neg\mathcal{B}_{\mathfrak{a}}(\mathfrak{b},\chi)$ are called belief literals of level 1, *the corresponding set is denoted by $\mathcal{B}\mathbf{Lit}_1(\mathfrak{a},\mathfrak{b})$. Let*

$$\mathcal{B}\mathbf{At}_1(\mathfrak{a},\mathfrak{A}) =_{def} \bigcup_{b\in\mathfrak{A}} \mathcal{B}\mathbf{At}_1(\mathfrak{a},\mathfrak{b}) \text{ and } \mathcal{B}\mathbf{Lit}_1(\mathfrak{a},\mathfrak{A}) =_{def} \bigcup_{b\in\mathfrak{A}} \mathcal{B}\mathbf{Lit}_1(\mathfrak{a},\mathfrak{b})$$

be the set of all $\mathfrak{a}$-belief atoms (resp. belief literals) relative to $\mathfrak{A}$. This reflects the idea that agent $\mathfrak{a}$ can have beliefs about **many** *agents in $\mathfrak{A}$.*

Example 9.1 (Belief Atoms In CFIT*)

- $\mathcal{B}_{\texttt{heli1}}(\texttt{tank1}, \mathbf{in}(\texttt{pos1}, \texttt{tank1} : \mathit{getPos}()))$
  This belief atom says that the agent, heli1 *believes that agent* tank1 *'s current state indicates that* tank1 *'s current position is* pos1*.*

- $\mathcal{B}_{\texttt{heli1}}(\texttt{tank1}, \mathbf{F}\mathit{attack}(\texttt{pos1}, \texttt{pos2}))$
  This belief atom says that the agent, heli1 *believes that agent* tank1 *'s current state indicates that it is forbidden for* tank1 *to attack from* pos1 *to* pos2*.*

- $\mathcal{B}_{\texttt{heli3}}(\texttt{tank1}, \mathbf{O}\mathit{drive}(\texttt{pos1}, \texttt{pos2}, 35))$
  This belief atom says that the agent, heli3 *believes that agent* tank1 *'s current state makes it obligatory for* tank1 *to drive from location* pos1 *to* pos2 *at* 35 *miles per hour.*

The language $\mathcal{B}\mathbf{Lit}_1(\mathfrak{a},\mathcal{A})$ does not allow agent $\mathfrak{a}$ to have beliefs of the form "*Agent* $\mathfrak{b}$ *believes that agent* $\mathfrak{c}$*'s state contains code call condition* χ," i.e., agent $\mathfrak{a}$ cannot express beliefs it has about the beliefs of another agent.

We introduce the following notation: for a given set X of formulae we denote by $\mathbf{Cl}_{\{\&,\neg\}}(X)$ the set of all conjunctions consisting of elements of X or their negations: $x_1 \wedge \neg x_2 \wedge \ldots \wedge x_n$, where $x_i \in X$. We emphasize that this does not correspond to the usual closure of X under & and $\neg$: in particular, it does not allow us to formulate disjunctions, if X consists of atoms.

Definition 9.2 (Nested Beliefs $\mathcal{B}\mathrm{Lit}_i(\mathbf{a},\mathbf{b})$, Belief language $\mathcal{BL}_i^{\mathbf{a}}$)
In the following let $\mathbf{a},\mathbf{b} \in \mathcal{A}$. *We want to define* $\mathcal{BL}_i^{\mathbf{a}}$, *the* belief language of agent $\mathbf{a}$ of level i. *This is done recursively as follows.*

$i \leq 1$: *In accordance with Definition 9.1 on page 407 (where we already defined* $\mathcal{B}\mathrm{At}_1(\mathbf{a},\mathbf{b})$) *we denote by* $\mathcal{B}\mathrm{At}_0(\mathbf{a},\mathbf{b})$ *as well as by* $\mathcal{B}\mathrm{Lit}_0(\mathbf{a},\mathbf{b})$

$$\{\phi \mid \phi \textit{ is a compatible code call condition or action atom}\}$$

the flat set of code call conditions or action atoms—no belief atoms are allowed. Furthermore, we define

$$\mathcal{BL}_0(\mathbf{a},\mathbf{b}) =_{def} \mathcal{B}\mathrm{At}_0(\mathbf{a},\mathbf{b})$$
$$\mathcal{BL}_1(\mathbf{a},\mathbf{b}) =_{def} \mathbf{Cl}_{\{\&,\neg\}}(\mathcal{B}\mathrm{At}_1(\mathbf{a},\mathbf{b})),$$

i.e., the set of formulae $\mathcal{B}\mathrm{At}_0(\mathbf{a},\mathbf{b})$, *resp. the of all conjunctions of belief literals from* $\mathcal{B}\mathrm{At}_1(\mathbf{a},\mathbf{b})$.

$$\mathcal{BL}_0^{\mathbf{a}} =_{def} \bigcup_{\mathbf{b}\in\mathcal{A}} \mathcal{BL}_0(\mathbf{a},\mathbf{b})$$
$$\mathcal{BL}_1^{\mathbf{a}} =_{def} \mathbf{Cl}_{\{\&,\neg\}}(\bigcup_{\mathbf{b}\in\mathcal{A}} \mathcal{BL}_1(\mathbf{a},\mathbf{b}))$$

are called the belief languages of agent $\mathbf{a}$ of level 0, resp. of level 1.

$\mathbf{i} > \mathbf{1}$: *To define nested belief literals we set for* $i > 1$

$$\begin{aligned}\mathcal{B}\mathbf{At}_i(\mathbf{a},\mathbf{b}) &=_{def} \{\mathcal{B}_{\mathbf{a}}(\mathbf{b},\beta) \,|\, \beta \in \mathcal{B}\mathbf{At}_{i-1}(\mathbf{b},\mathcal{A})\},\\ \mathcal{B}\mathbf{Lit}_i(\mathbf{a},\mathbf{b}) &=_{def} \{(\neg)\mathcal{B}_{\mathbf{a}}(\mathbf{b},\beta) \,|\, \beta \in \mathcal{B}\mathbf{At}_{i-1}(\mathbf{b},\mathcal{A})\}.\end{aligned}$$

This finishes the recursive definition of $\mathcal{B}\mathbf{Lit}_i(\mathbf{a},\mathbf{b})$.

The definition of the belief language of agent $\mathbf{a}$ of level i *is:*

$$\mathcal{BL}_i^{\mathbf{a}} =_{def} \mathbf{Cl}_{\{\&,\neg\}}(\bigcup_{\mathbf{b}\in\mathcal{A}} \mathcal{BL}_i(\mathbf{a},\mathbf{b})) \tag{9.4}$$

$$\mathcal{BL}_i(\mathbf{a},\mathbf{b}) =_{def} \mathbf{Cl}_{\{\&,\neg\}}(\mathcal{B}\mathbf{At}_i(\mathbf{a},\mathbf{b})).$$

Finally we define the maximal belief language an agent $\mathfrak{a}$ *can have:*

$$\mathcal{BL}^{\mathfrak{a}}_{\infty} =_{def} \mathbf{Cl}_{\{\&,\neg\}}(\bigcup_{i=0}^{\infty} \mathcal{BL}^{\mathfrak{a}}_{i}). \tag{9.5}$$

Formulae in this language are also called general belief formulae.

We will later also use the following definitions:

1. $\mathcal{B}\mathbf{At}_i(\mathfrak{a},\mathfrak{A}) =_{def} \bigcup_{\mathfrak{b}\in\mathfrak{A}} \mathcal{B}\mathbf{At}_i(\mathfrak{a},\mathfrak{b})$ *is called the set of belief atoms of depth* i.
2. $\mathcal{B}\mathbf{Lit}_i(\mathfrak{a},\mathfrak{A}) =_{def} \bigcup_{\mathfrak{b}\in\mathfrak{A}} \mathcal{B}\mathbf{Lit}_i(\mathfrak{a},\mathfrak{b})$ *is called the set of belief literals of depth* i.
3. *We define*

$$\mathcal{B}\mathbf{At}_{\infty}(\mathfrak{a},\mathfrak{A}) =_{def} \bigcup_{i=0}^{\infty} \mathcal{B}\mathbf{At}_i(\mathfrak{a},\mathfrak{A}), \quad \mathcal{B}\mathbf{Lit}_{\infty}(\mathfrak{a},\mathfrak{A}) =_{def} \bigcup_{i=0}^{\infty} \mathcal{B}\mathbf{Lit}_i(\mathfrak{a},\mathfrak{A}).$$

Why is it so complicated?

A nested belief atom of the form

$$\mathcal{B}_{a}(b, \mathcal{B}_{c}(d, \chi))$$

does not make sense (because $b \neq c$).

Thus every agent keeps track of only its *own* beliefs, not those of other agents!!

- The closure under $\{\&, \neg\}$ in Equation (9.4 on page 411) allows us to use conjunctions with respect to different agents $\mathcal{B}_{a}(b, \chi) \wedge \mathcal{B}_{a}(c, \chi')$.
- The closure in Equation (9.5 on page 411) allows us to also use different nested levels of beliefs, like $\mathcal{B}_{a}(b, \chi) \wedge \mathcal{B}_{a}(c, \mathcal{B}_{c}(d, \chi'))$.

Example 9.2 (Belief Formulae for CFIT*)

The following are belief formulae from $\mathcal{BL}_1^{\mathtt{heli1}}$, $\mathcal{BL}_2^{\mathtt{tank1}}$ and $\mathcal{BL}_3^{\mathtt{coord}}$.

- $\mathcal{B}_{\mathtt{heli1}}(\mathtt{tank1}, \mathbf{in}(\mathtt{pos1}, \mathtt{tank1}: \mathit{getPosition}())).$
  This formula is in $\mathcal{BL}_1^{\mathtt{heli1}}$. *It says that agent* **heli1** *believes that agent* **tank1** *'s current state indicates that* **tank1** *'s current position is* `pos1`.

- $\mathcal{B}_{\mathtt{tank1}}(\mathtt{heli1}, \mathcal{B}_{\mathtt{heli1}}(\mathtt{tank1}, \mathbf{in}(\mathtt{pos1}, \mathtt{tank1}: \mathit{getPosition}()))).$
  This formula is in $\mathcal{BL}_2^{\mathtt{tank1}}$. *It says that agent* **tank1** *believes that agent* **heli1** *believes that agent* **tank1** *'s current position is* `pos1`.

- $\mathcal{B}_{\mathtt{coord}}(\mathtt{tank1}, \mathcal{B}_{\mathtt{tank1}}(\mathtt{heli1}, \mathcal{B}_{\mathtt{heli1}}(\mathtt{tank2}, \mathbf{in}(\mathtt{pos2}, \mathtt{tank2}: \mathit{getPosition}())))).$
  This formula is in $\mathcal{BL}_3^{\mathtt{coord}}$. *It says that agent* **coord** *believes that agent* **tank1** *believes that* **heli1** *believes that agent* **tank2** *'s current position is* `pos2`.

However, the following formula does not belong to any of the above belief languages:

$$\mathcal{B}_{\mathtt{tank1}}(\mathtt{heli1}, \mathcal{B}_{\mathtt{tank1}}(\mathtt{tank1}, \mathbf{in}(\mathtt{pos1}, \mathtt{tank}:\mathit{getPosition}()))).$$

The reason for this is because in heli1's state there can be no beliefs belonging to tank1.

9.2.1 Basic Belief Table

Definition 9.3 (Basic Belief Table $\mathbf{BBT}^{\mathfrak{a}}$)
Every agent $\mathfrak{a}$ *has an associated* basic belief table $\mathbf{BBT}^{\mathfrak{a}}$ *which is a set of pairs*

$$\langle \mathfrak{h}, \phi \rangle$$

where $\mathfrak{h} \in \mathcal{A}$ *and* $\phi \in \mathcal{BL}_i^{\mathfrak{h}}$, $i \in \mathbb{N}$.

For example, if the entry $\langle \mathfrak{b}, \mathcal{B}_{\mathfrak{b}}(\mathfrak{a}, \chi) \rangle$ is in the table $\mathbf{BBT}^{\mathfrak{a}}$, then this intuitively means that agent $\mathfrak{a}$ believes that agent $\mathfrak{b}$ has the code call condition χ among its own beliefs about agent $\mathfrak{a}$. Here $\phi \in \mathcal{BL}_1^{\mathfrak{b}}$.

Example 9.3 (Basic Belief Table for CFIT* Agents)
We define suitable basic belief tables for agent tank1 *(Table 9.1) and* heli1 *(Table 9.2).*

Agent	*Formula*
heli1	$\mathbf{in}(\texttt{pos1}, \mathtt{heli1} : \textit{getPosition}())$
heli2	$\mathcal{B}_{\mathtt{heli2}}(\mathtt{tank1}, \mathbf{in}(\texttt{pos1}, \mathtt{tank1} : \textit{getPosition}()))$
tank2	$\mathcal{B}_{\mathtt{tank2}}(\mathtt{heli1}, \mathcal{B}_{\mathtt{heli1}}(\mathtt{tank1}, \mathbf{in}(\texttt{pos3}, \mathtt{tank1} : \textit{getPosition}())))$

Table 9.1: A Basic Belief Table for agent tank1.

Agent	*Formula*
heli2	$\mathbf{in}(\mathtt{pos2}, \mathsf{heli2} : \mathit{getPosition}())$
tank1	$\mathbf{in}(\mathtt{pos1}, \mathsf{tank1} : \mathit{getPosition}())$
tank1	$\mathcal{B}_{\mathtt{tank1}}(\mathsf{heli1}, \mathbf{in}(\mathtt{pos1}, \mathsf{heli1} : \mathit{getPosition}()))$
tank2	$\mathcal{B}_{\mathtt{tank2}}(\mathsf{tank1}, \mathcal{B}_{\mathtt{tank1}}(\mathsf{heli1}, \mathbf{in}(\mathtt{pos4}, \mathsf{heli1} : \mathit{getPosition}())))$

Table 9.2: A Basic Belief Table for agent heli1.

These tables indicate that tank1 *and* heli1 *work closely together and know their positions. Both believe that the other knows about both positions.* tank1 *also believes that* tank2 *believes that in* heli2*'s state,* tank1 *is in position* `pos3` *(which is actually wrong).*

heli1 *thinks that* tank2 *believes that* tank1 *believes that* heli1 *is in position* `pos4`*, which is also wrong.*

What kind of operations should we support on belief tables? We distinguish between two different types:

1. For a given agent h, other than a, we may want to select all entries in the table having h as first argument.
2. For a given belief formula ϕ, we may be interested in all those entries, whose second argument “implies” (w.r.t. some underlying definition of entailment) the given formula ϕ.

9.2.2 Belief Semantics Table

Agent $\mathfrak{a}$ may associate different background theories with different agents: it may assume that agent $\mathfrak{h}$ reasons according to semantics $\mathcal{B}Sem^{\mathfrak{a}}_{\mathfrak{h}}$ and assumes that agent $\mathfrak{h}'$ adopts a stronger semantics $\mathcal{B}Sem^{\mathfrak{a}}_{\mathfrak{h}'}$. We will store the information in a separate relational data structure:

Definition 9.4 (Belief Semantics Table BSemT$^{\mathfrak{a}}$ of Agent $\mathfrak{a}$)

Every agent $\mathfrak{a}$ *has an associated* belief semantics table **BSemT**$^{\mathfrak{a}}$ *which is a set of pairs*

$$\langle \mathfrak{h}, \mathcal{BSem}^{\mathfrak{a}}_{\mathfrak{h}} \rangle$$

where $\mathfrak{h} \in \mathfrak{A}$, $\mathcal{BSem}^{\mathfrak{a}}_{\mathfrak{h}}$ *is a belief semantics over* $\mathcal{BL}^{\mathfrak{h}}_{i}$ *and* $i \in \mathbb{N}$ *is fixed. In addition we require at most one entry per agent* $\mathfrak{h}$*. Hence,* $\mathcal{BSem}^{\mathfrak{a}}_{\mathfrak{h}}$ *determines an entailment relation*

$$\phi \models_{\mathcal{BSem}^{\mathfrak{a}}_{\mathfrak{h}}} \psi$$

between belief formulae $\phi, \psi \in \mathcal{BL}^{\mathfrak{h}}_{i}$*. We also assume the existence of the following function (which constitutes an extended code call, see Definition 9.11 on page 449) over* **BSemT**$^{\mathfrak{a}}$*:*

$$\mathbf{BSemT}^{\mathfrak{a}} : \mathrm{select}(\mathrm{agent}, =, \mathfrak{h}),$$

which selects all entries corresponding to a specific agent $\mathfrak{h} \in \mathfrak{A}$.

Example 9.4 (Belief Semantics Tables for CFIT* Agents)

We briefly describe what suitable Belief Semantics Table for heli1 *and* tank1 *may look like. We have to define entailment relations* $\mathcal{B}Sem^{\mathtt{tank1}}_{\mathtt{tank2}}$, $\mathcal{B}Sem^{\mathtt{tank1}}_{\mathtt{heli1}}$, $\mathcal{B}Sem^{\mathtt{tank1}}_{\mathtt{heli2}}$ *, and* $\mathcal{B}Sem^{\mathtt{heli1}}_{\mathtt{tank1}}$, $\mathcal{B}Sem^{\mathtt{heli1}}_{\mathtt{tank2}}$, $\mathcal{B}Sem^{\mathtt{heli1}}_{\mathtt{heli2}}$. *For simplicity we restrict these entailment relations to belief formulae of level at most* 1*, i.e.,* $\mathcal{BL}^{\mathtt{h}}_1$.

1. $\mathcal{B}Sem^{\mathtt{heli1}}_{\mathtt{tank1}}$*: The smallest entailment relation satisfying the schema*

$$\mathcal{B}_{\mathtt{tank1}}(\mathtt{tank1.1},\chi) \to \chi.$$

This says that heli1 *believes that all beliefs of* tank1 *about* tank1.1 *are actually true:* tank1 *knows all about* tank1.1.

2. $\mathcal{B}Sem^{\mathtt{heli1}}_{\mathtt{tank2}}$*: The smallest entailment relation satisfying the schema*

$$\mathcal{B}_{\mathtt{tank2}}(\mathtt{tank2.1},\chi) \to \chi.$$

This says that heli1 *believes that all beliefs of* tank2 *about* tank2.1 *are actually true:* tank2 *knows all about* tank2.1.

3. *$\mathcal{B}Sem^{\mathtt{tank1}}_{\mathtt{heli1}}$: The smallest entailment relation satisfying the schema*

$$\mathcal{B}_{\mathtt{heli1}}(\mathtt{tank1},\chi) \rightarrow \chi.$$

This says that tank1 *believes that if* heli1 *believes in χ for* tank1*, then this is true (*heli1 *knows all about* tank1*. An instance of χ is*
in(`pos1`, tank1 : ***getPosition()*****)**.

4. *$\mathcal{B}Sem^{\mathtt{tank1}}_{\mathtt{heli2}}$: The smallest entailment relation satisfying the schema*

$$\mathcal{B}_{\mathtt{heli2}}(\mathtt{tank2},\chi) \wedge \mathcal{B}_{\mathtt{heli2}}(\mathtt{tank2.1},\chi) \rightarrow \chi.$$

This says that tank1 *believes that if* heli2 *believes that χ is true both for* tank2 *and* tank2.1 *then this is actually true.*

The notion of a semantics used in the belief semantics table is very general: it can be an arbitrary relation on $\mathcal{BL}_i^{\mathfrak{h}} \times \mathcal{BL}_i^{\mathfrak{h}}$.

As an example, consider the following two simple axioms that can be built into a semantics:

$$\begin{array}{lccc} (1) & \mathcal{B}_{\mathfrak{h}_2}(\mathfrak{h}, \chi) & \Rightarrow & \mathcal{B}_{\mathfrak{h}_2}(\mathfrak{h}', \chi) \\ (2) & \mathcal{B}_{\mathfrak{h}_2}(\mathfrak{h}, \chi) & \Rightarrow & \chi \end{array}$$

The first axiom refers to different agents $\mathfrak{h}, \mathfrak{h}'$ while the second combines different *levels* of belief atoms: see Equations 9.4 on page 411 and 9.5 on page 411 and the discussion after Definition 9.2 on page 410. In many applications, however, such axioms will not occur: $\mathfrak{h} = \mathfrak{h}'$ is fixed and the axioms operate on the same level i of belief formulae.

Suppose an agent $\mathfrak{a}$ believes that another agent $\mathfrak{h}_1$ reasons according to the feasible semantics, $\mathfrak{h}_2$ reasons according to the rational semantics etc. It would be nice if this could be encoded as follows in $\mathbf{BSemT}^{\mathfrak{a}}$

$$\langle \mathfrak{h}_1, \mathsf{Sem}_{feas} \rangle$$
$$\langle \mathfrak{h}_2, \mathsf{Sem}_{rat} \rangle$$
$$\langle \mathfrak{h}_3, \mathsf{Sem}_{reas} \rangle$$

Remark 1 shows that this is indeed possible.

The idea is to use the semantics Sem of the action program $\mathcal{P}^{\mathfrak{a}}(\mathfrak{b})$ (that $\mathfrak{a}$ believes $\mathfrak{b}$ to have) for the evaluation of the belief formulae.

Remark 1 (Sem for Agent Programs induces $\mathcal{B}Sem^{\mathfrak{a}}_{\mathfrak{h}}$) *Let Sem be the* reasonable, rational *or* feasible *semantics for agent programs (i.e., not containing beliefs). Suppose agent $\mathfrak{a}$ believes that agent $\mathfrak{h}$ reasons according to Sem. Let $\mathcal{P}(\mathfrak{h})$ be the agent program of $\mathfrak{h}$ and $\mathcal{O}(\mathfrak{h})$, $\mathcal{AC}(\mathfrak{h})$ and $\mathcal{IC}(\mathfrak{h})$ the state, action constraints and integrity constraints of $\mathfrak{h}$. Then there is a basic belief table* **BSemT**$^{\mathfrak{a}}$ *and a belief semantics $\mathcal{B}Sem^{\mathfrak{a}}_{\mathfrak{h}}$ induced by Sem such that*

- *$\mathfrak{a}$ believes in $\mathfrak{h}$'s state, and*
- *$\mathfrak{a}$ believes in all actions taken by $\mathfrak{h}$ with respect to Sem and $\mathcal{P}(\mathfrak{h})$.*

More generally: let $i \in \mathbb{N}$ *and suppose agent* $\mathfrak{a}$ *believes that agent* $\mathfrak{h}_1$ *believes that agent* $\mathfrak{h}_2$ *believes that* $\ldots$ *believes that agent* $\mathfrak{h}_{i-1}$ *acts according to* $\mathcal{P}^{\mathfrak{a}}(\sigma)$ *(where* $\sigma =_{def} [\mathfrak{h}_1, \mathfrak{h}_2, \ldots, \mathfrak{h}_{i-1}]$*) and state* $O(\sigma)$[a]*. Then there is a basic belief table* **BSemT**$^{\mathfrak{a}}$ *and a belief semantics* $\mathcal{B}Sem^{\mathfrak{a}}_{\sigma}$ *induced by* **Sem** *on a suitably restricted subset of* $\mathcal{BL}^{\mathfrak{h}}_1 \times \mathcal{BL}^{\mathfrak{h}}_1$ *such that*

- $\mathfrak{a}$ *believes in* $\mathfrak{h}_{i-1}$*'s state, and*
- $\mathfrak{a}$ *believes in all actions taken by* $\mathfrak{h}_{i-1}$ *with respect to* **Sem** *and* $\mathcal{P}(\sigma)$.

[a]See Definition 9.10 on page 448 and Definition 9.8 on page 439 for a detailed introduction of these concepts.

9.2.3 Belief Tables

We are now ready to give the full definition of a belief table.

Definition 9.5 (Belief Table $\mathbf{BT}^{a}$)
Every agent a *has an associated* belief table $\mathbf{BT}^{a}$, *which consists of triples*

$$\langle h, \phi, \chi_{\mathcal{B}} \rangle$$

where $h \in A$, $\phi \in \mathcal{BL}_i^{h}$ *and* $\chi_{\mathcal{B}} \in \mathcal{B}Cond^{a}(h)$ *is a* belief condition of a.

The role of such a belief condition is to extend the expressiveness of the basic belief table by restricting the applicability to particular states, namely those satisfying the belief condition. Intuitively, $\langle b, \phi, \chi_{\mathcal{B}} \rangle$ *means that*

Agent a *believes that* ϕ *is true in agent* b*'s state, if the condition* $\chi_{\mathcal{B}}$ *holds.*

$\mathbf{BT}^{a}$ *and* $\mathbf{BSemT}^{a}$, *taken together,* simulate *agent* b*'s state as believed by agent* a.

We also assume the existence of the following two functions over $\mathbf{BT}^{\mathfrak{a}}$*:*

$$\mathbf{BT}^{\mathfrak{a}} : \text{proj-select}(\text{agent}, =, \mathfrak{h})$$

which selects all entries of $\mathbf{BT}^{\mathfrak{a}}$ *of the form* $\langle \mathfrak{h}, \phi, \mathbf{true} \rangle$ *(i.e., corresponding to a specific agent* $\mathfrak{h} \in \mathfrak{A}$ *and having the third entry empty) and projects them, and*

$$\mathbf{BT}^{\mathfrak{a}} : \text{B-proj-select}(r, \mathfrak{h}, \phi)$$

for all $r \in \mathcal{R} =_{def} \{\Rightarrow, \Leftarrow, \Leftrightarrow\}$ *and for all belief formulae* $\phi \in \mathcal{BL}^{\mathfrak{h}}_{\infty}$*. This function selects all entries of* $\mathbf{BT}^{\mathfrak{a}}$ *of the form* $\langle \mathfrak{h}, \psi, \mathbf{true} \rangle$ *that contain a belief formula* ψ *which is in relation* r *to* ϕ *with respect to the semantics* $\mathcal{BSem}^{\mathfrak{a}}_{\mathfrak{h}}$ *as specified in the belief semantics table* $\mathbf{BSemT}^{\mathfrak{a}}$ *and projects them on the first two arguments.*

For example, if we choose $\Rightarrow \in \mathcal{R}$ *as the relation* r *then*

$$(\psi \Rightarrow \phi) \in \mathcal{BSem}^{\mathfrak{a}}_{\mathfrak{h}} \text{ or, equivalently, } \models_{\mathcal{BSem}^{\mathfrak{a}}_{\mathfrak{h}}} (\psi \Rightarrow \phi) \text{ says}$$

ϕ *is entailed by* ψ *relative to semantics* $\mathcal{BSem}^{\mathfrak{a}}_{\mathfrak{h}}$*.*

A belief condition $\chi_{\mathcal{B}}$ that occurs in an entry $\langle \mathbf{b}, \phi, \chi_{\mathcal{B}} \rangle$ must be evaluated in what agent $\mathbf{a}$ believes is agent $\mathbf{b}$'s state.

This is important because the code call conditions must be compatible and therefore not only depend on agent $\mathbf{a}$ but also on agent $\mathbf{b}$.

Agent	**Formula**	**Condition**
heli1	$\mathbf{in}(\texttt{pos1}, \mathbf{heli1} : \mathit{getPosition}())$	**true**
heli2	$\mathcal{B}_{\mathtt{heli2}}(\mathbf{tank1}, \mathbf{in}(\texttt{pos1}, \mathbf{tank1} : \mathit{getPosition}()))$	$\mathcal{B}cond_1^{\mathtt{tank1}}$
tank2	$\mathcal{B}_{\mathtt{tank2}}(\mathbf{heli1}, \mathcal{B}_{\mathtt{heli1}}(\mathbf{tank1}, \mathbf{in}(\texttt{pos3}, \mathbf{tank1} : \mathit{getPosition}())))$	$\mathcal{B}cond_2^{\mathtt{tank1}}$

Table 9.3: A Belief Table for agent tank1.

Example 9.5 (Belief Table for CFIT* Agents Revisited)

We now consider Table 9.3 on the page before and extend our basic belief tables for agent **tank1** *(Table 9.1 on page 417) and* **heli1** *(Table 9.2 on page 417). Let* $\mathcal{B}cond_1^{\mathtt{tank1}}$ *be the code call condition* $\mathbf{in}(\mathtt{pos1}, \mathtt{tank1} : getPosition())$ *and define* $\mathcal{B}cond_2^{\mathtt{tank1}}$ *by*

$$\mathbf{in}(\langle \mathtt{heli1}, belief\ atom\rangle, \mathbf{BT}^{a} : \text{proj-select}(\text{agent}, =, \mathtt{heli1})),$$

where

$$belief\ atom =_{def} \mathcal{B}_{\mathtt{heli1}}(\mathtt{tank1}, \mathbf{in}(\mathtt{pos3}, \mathtt{tank1} : getPosition())).$$

The first row in the table says that **tank1** *unconditionally believes that in* **heli1***'s state the position for* **heli1** *is* `pos1`.

The second row in the belief table above, says that **tank1** *believes that if* **tank1***'s position is* `pos1`, **heli2** *believes that in* **tank1***'s state the position of* **tank1** *is* `pos1`.

The third row in the belief table says that if **tank1** *believes* **heli1** *believes that* **tank1** *'s position is* `pos3`*, then* **tank2** *believes* **heli1** *believes* **tank1** *'s position is* `pos3`.

The table for **heli1** *is as shown in Table 9.4, where* $\mathcal{B}cond_1^{\mathtt{heli1}}$ *stands for*

$$\mathbf{in}(\mathtt{pos2}, \mathtt{heli2} : \mathit{getPosition()})$$

and $\mathcal{B}cond_2^{\mathtt{tank1}}$ *is defined by*

$$\mathbf{in}(\langle \mathtt{tank1}, \mathit{belief\ atom}\rangle, \mathbf{BT}^{\mathbf{a}} : \text{proj-select}(\text{agent}, =, \mathtt{tank1})),$$

where

$$\mathit{belief\ atom} =_{def} \mathcal{B}_{\mathtt{tank1}}(\mathtt{heli1}, \mathbf{in}(\mathtt{pos4}, \mathtt{heli1} : \mathit{getPosition()})).$$

Agent	*Formula*	*Condition*
heli2	in(pos2, heli2 : *getPosition()*)	**true**
tank1	in(pos1, tank1 : *getPosition()*)	**true**
tank1	$\mathcal{B}_{\text{tank1}}$(heli1, in(pos1, heli1 : *getPosition()*))	$\mathcal{B}cond_1^{\text{heli1}}$
tank2	$\mathcal{B}_{\text{tank2}}$(tank1, $\mathcal{B}_{\text{tank1}}$(heli1, in(pos4, heli1 : *getPosition()*)))	$\mathcal{B}cond_2^{\text{heli1}}$

Table 9.4: A Belief Table for agent heli1.

9.3 Meta Agent Programs and Status Sets

Definition 9.6 (Meta Agent Program (map) $\mathcal{BP}$)
A meta agent rule, *(mar for short), for agent* $\mathfrak{a}$ *is an expression r of the form*

$$\mathsf{Op}\,\alpha(\vec{t}) \leftarrow L_1, \ldots, L_n \tag{9.6}$$

where $\mathsf{Op}\,\alpha(\vec{t})$ *is an action status atom, and each of* $L_1, \ldots, L_n$ *is either a code call literal, an action literal or a belief literal from* $\mathcal{B}\mathbf{Lit}_\infty(\mathfrak{a}, \mathcal{A})$*.*

A meta agent program, *(map for short), for agent* $\mathfrak{a}$ *is a finite set* $\mathcal{BP}$ *of meta agent rules for* $\mathfrak{a}$*.*

Example 9.6 (map's For CFIT*-Agents)

Let heli1 *'s meta agent program be as follows:*

$$\begin{aligned}\mathbf{P}\ \textit{attack}(\mathtt{P1},\mathtt{P2}) \ \leftarrow\ & \mathcal{B}_{\mathtt{heli1}}(\mathtt{tank1}, \mathbf{in}(\mathtt{P2}, \mathtt{tank1}:\textit{getPos}())),\\ & \mathbf{P}\ \textit{fly}(\mathtt{P1},\mathtt{P3},\mathtt{A},\mathtt{S}),\\ & \mathbf{P}\ \textit{attack}(\mathtt{P3},\mathtt{P2}).\end{aligned}$$

where *attack*(P1,P2) *is an action which means attack position* P2 *from position* P1. heli1 *'s program says* heli1 *can attack position* P2 *from* P1 *if* heli1 *believes* tank1 *is in position* P2, heli1 *can fly from* P1 *to another position* P3 *at altitude* A *and speed* S, *and* heli1 *can attack position* P2 *from* P3.

Let tank1 *'s meta agent program be as follows:*

$$\begin{aligned}\mathbf{O}\ \textit{attack}(\mathtt{P1},\mathtt{P2}) \ \leftarrow\ & \mathbf{O}\ \textit{driveRoute}([\mathtt{P0},\mathtt{P1},\mathtt{P2},\mathtt{P3}],\mathtt{S}),\\ & \mathcal{B}_{\mathtt{tank1}}(\mathtt{tank2}, \mathbf{in}(\mathtt{P2}, \mathtt{tank2}:\textit{getPos}())).\end{aligned}$$

If tank1 *must drive through a point where it believes* tank2 *is, it must attack* tank2.

From now on we assume that the software package $\mathcal{S}^{\mathfrak{a}} = (\mathcal{T}_{\mathcal{S}^{\mathfrak{a}}}, \mathcal{F}_{\mathcal{S}^{\mathfrak{a}}})$ of each agent $\mathfrak{a}$ contains as distinguished data types

1. the belief table $\mathbf{BT}^{\mathfrak{a}}$, and
2. the belief semantics table $\mathbf{BSemT}^{\mathfrak{a}}$,

as well as the corresponding functions

$$\mathbf{BT}^{\mathfrak{a}} : \text{B-proj-select}(r, \mathfrak{h}, \phi) \text{ and } \mathbf{BSemT}^{\mathfrak{a}} : \text{select}(\text{agent}, =, \mathfrak{h}).$$

What is a status set?

Definition 9.7 (Belief Status Set $\mathcal{BS}$)
A belief status set $\mathcal{BS}$ of agent $\mathfrak{a}$, also written $\mathcal{BS}(\mathfrak{a})$, is a set consisting of two kinds of elements:

- *ground action status atoms over $\mathcal{S}^{\mathfrak{a}}$ and*
- *belief atoms from $\mathcal{B}\mathbf{At}_{\infty}(\mathfrak{a}, \mathbf{A})$ of level greater or equal to 1.*

The reason that we do not allow belief atoms of level 0 is to avoid having code call conditions in our set. In agent programs without beliefs (which we want to extend) they are not allowed (see Definition 7.15 on page 286).

We note that such a status set must be determined in accordance with

1. the map $\mathcal{BP}$ of agent $\mathbf{a}$,
2. the current state $\mathcal{O}$ of $\mathbf{a}$,
3. the underlying set of action ($\mathcal{AC}$) and integrity constraints ($\mathcal{IC}$) of $\mathbf{a}$.

- In contrast to agent programs without beliefs we now have **to cope with all agents about which $\mathbf{a}$ holds certain beliefs**.
- Even if the map $\mathcal{BP}$ does not contain nested beliefs (which are allowed), we cannot restrict ourselves to belief atoms of level 1. This is because the belief table $\mathbf{BT}^{\mathbf{a}}$ may contain nested beliefs and, by the belief semantics table $\mathbf{BSemT}^{\mathbf{a}}$, such nested beliefs may imply (trigger) other beliefs.

Any belief status set $\mathcal{BS}$ of agent $\mathfrak{a}$ induces, in a natural way, for any agent $\mathfrak{b} \in \mathfrak{A}$, two sorts of sets: the *state* and the various *action status sets* that agent $\mathfrak{a}$ believes other agents $\mathfrak{b}$ to hold or those that $\mathfrak{a}$ believes other agents $\mathfrak{b}$ to hold about other agents $\mathfrak{c}$. To easily formalize the latter conditions, we introduce the notion of a sequence:

Definition 9.8 (Sequence σ, $[\rho]$ of Agents)
A sequence σ of agents from $\mathfrak{A}$ is defined inductively as follows:

1. *The empty sequence $[]$ is a sequence.*
2. *If $\mathfrak{a} \in \mathfrak{A}$ and $[\rho]$ is a sequence, then $[\mathfrak{a}]$, $[-\mathfrak{a}]$, $[\mathfrak{a},\rho]$, $[\rho,\mathfrak{a}]$ are sequences.*

We use both σ and $[\rho]$ to refer to an arbitrary sequence.

- The overall intuition of the formula $\mathcal{B}_{\mathfrak{a}}(\mathfrak{b}, \mathcal{B}_{\mathfrak{b}}(\mathfrak{c}, \mathcal{B}_{\mathfrak{c}}(\mathfrak{d}, \chi)))$ is that if we keep agent $\mathfrak{a}$ in mind, then agent $\mathfrak{a}$ believes in a code call condition of type $[\mathfrak{b}, \mathfrak{c}, \mathfrak{d}]$, i.e., a ccc that $\mathfrak{b}$ believes that $\mathfrak{c}$ believes that it holds in $\mathfrak{d}$'s state.
- We also say sometimes "σ's state" and refer to the code call conditions that are true in what $\mathfrak{a}$ believes that $\mathfrak{b}$ believes ... where $[\mathfrak{a}, \mathfrak{b}, \dots] = \sigma$.

Definition 9.9 (Induced Status Set $\Pi_{\mathbf{b}}^{\text{action}}(\mathcal{BS})$ and State $\Pi_{\mathbf{b}}^{\text{state}}(\mathcal{BS})$)
Let $\mathbf{a}, \mathbf{b}$ be agents and $\mathcal{BP}$ a map of $\mathbf{a}$. Every belief status set $\mathcal{BS}$ of an agent $\mathbf{a}$ induces the following two sets describing $\mathbf{a}$'s beliefs about $\mathbf{b}$'s actions and $\mathbf{b}$'s state

$$\begin{array}{lcllll} \Pi_{\mathbf{b}}^{\text{action}}(\mathcal{BS}) & =_{def} & \{ & Op\,\alpha(\vec{t}) & \mid & \mathcal{B}_{\mathbf{a}}(\mathbf{b}, Op\,\alpha(\vec{t})) \in \mathcal{BS},\ \textit{where } Op \in \{\mathbf{O}, \mathbf{W}, \mathbf{P}, \mathbf{F}, \mathbf{Do}\}\} \\ \Pi_{\mathbf{b}}^{\text{state}}(\mathcal{BS}) & =_{def} & \{ & \chi & \mid & \mathcal{B}_{\mathbf{a}}(\mathbf{b}, \chi) \in \mathcal{BS} \textit{ and } \chi \textit{ is a code call condition}\} \end{array}$$

Now assume that agent $\mathbf{a}$ believes in $\mathcal{BS}$. Then $\Pi_{\mathbf{b}}^{\text{state}}(\mathcal{BS})$ formalizes the state of agent $\mathbf{b}$ as believed by agent $\mathbf{a}$. Similarly, $\Pi_{\mathbf{b}}^{\text{action}}(\mathcal{BS})$ represents the action status set of agent $\mathbf{b}$ as believed by agent $\mathbf{a}$.

For any sequence $\boldsymbol{\sigma}$, $\mathcal{BS}$ induces the following two sets:

$\Pi_{\boldsymbol{\sigma}}^{\text{action}}(\mathcal{BS})$	*describing $\mathbf{a}$'s belief about actions corresponding to $\boldsymbol{\sigma}$*
$\Pi_{\boldsymbol{\sigma}}^{\text{state}}(\mathcal{BS})$	*describing $\mathbf{a}$'s belief about the state corresponding to $\boldsymbol{\sigma}$,*

depending on the depth of the belief atoms occuring in $\mathcal{BP}$.

The formal definition of $\Pi^{\text{action}}_{\sigma}(\mathcal{BS})$ and $\Pi^{\text{state}}_{\sigma}(\mathcal{BS})$ is by induction on the structure of $\boldsymbol{\sigma}$. As it should be clear, we avoid this technical definition. Instead we shortly illustrate the case for $\boldsymbol{\sigma} = [\mathbf{b}, \mathbf{c}]$. Then

$$\begin{array}{lllll}
\Pi^{\text{action}}_{[\mathbf{b},\mathbf{c}]}(\mathcal{BS}) & =_{def} & \{ & \mathsf{Op}\,\alpha(\vec{t}) & \mid \boldsymbol{\mathcal{B}}_{\mathbf{a}}(\mathbf{b}, \boldsymbol{\mathcal{B}}_{\mathbf{b}}(\mathbf{c}, \mathsf{Op}\,\alpha(\vec{t}))) \in \mathcal{BS}\} \\
\Pi^{\text{state}}_{[\mathbf{b},\mathbf{c}]}(\mathcal{BS}) & =_{def} & \{ & \chi & \mid \boldsymbol{\mathcal{B}}_{\mathbf{a}}(\mathbf{b}, \boldsymbol{\mathcal{B}}_{\mathbf{b}}(\mathbf{c}, \chi)) \in \mathcal{BS}\}.
\end{array}$$

- It is important to note that for any sequence, σ of agents,
  - $\Pi_{\sigma}^{\text{action}}(\mathcal{BS})$ is a set of action status atoms.
  - $\Pi_{\sigma}^{\text{state}}(\mathcal{BS})$ is a set of code call conditions that do *not* involve beliefs.
- For the empty sequence $[]$, we identify $\Pi_{[]}^{\text{action}}(\mathcal{BS})$ (resp. $\Pi_{[]}^{\text{state}}(\mathcal{BS})$) with a's own action status set (resp. a's own state) as defined by the subset of $\mathcal{BS}$ not involving belief atoms.

Example 9.7 (Belief Status Sets for CFIT*-Agents)

We consider the map of heli1 *given in Example 9.6 on page 435*

$$\begin{aligned}\mathcal{BS}(\mathtt{heli1}) =_{def} \{ \; & \mathbf{P}\mathit{fly}(\mathtt{pointA}, \mathtt{pointB}, 10000, 200), \mathbf{O}\mathit{fly}(\mathtt{pointA}, \mathtt{pointB}, 10000, 200),\\ & \mathcal{B}_{\mathtt{heli1}}(\mathtt{heli2}, \mathbf{P}\mathit{fly}(\mathtt{PointA}, \mathtt{PointB}, 10000, 200)),\\ & \mathcal{B}_{\mathtt{heli1}}(\mathtt{heli2}, \mathbf{in}(\mathtt{pos}, \mathtt{heli2} : \mathit{getPos}())),\\ & \mathcal{B}_{\mathtt{heli1}}(\mathtt{heli2}, \mathcal{B}_{\mathtt{heli2}}(\mathtt{tank1}, \mathbf{in}(\mathtt{pos}, \mathtt{tank1} : \mathit{getPos}())))\\ & \mathcal{B}_{\mathtt{heli1}}(\mathtt{heli2}, \mathcal{B}_{\mathtt{heli2}}(\mathtt{tank1}, \mathbf{P}\mathit{drive}(\mathtt{pointX}, \mathtt{pointY}, 40)))\}\end{aligned}$$

This belief status set is for heli1 *and it says:*

1. *It is possible to fly from pointA to pointB at an altitude of 10000 feet and a speed of 200 knots.*
2. *It is obligatory to fly from pointA to pointB at an altitude of 10000 feet and a speed of 200 knots.*

3. **heli1** *believes that in* **heli2***'s state it is possible to fly from pointA to pointB at 10000 feet and 200 knots.*
4. **heli1** *believes that in* **heli2***'s state the position of* **heli2** *is* `pos`.
5. **heli1** *believes* **heli2** *believes that* **tank1***'s position is* `pos`.
6. **heli1** *believes* **heli2** *believes that in* **tank1***'s state it is possible to drive from* `pointX` *to* `pointY` *at 40 miles per hour.*

We then have:

$$\Pi^{\text{action}}_{\mathbf{heli2}}(\mathcal{BS}(\mathbf{heli1})) = \{\mathbf{P}\mathit{fly}(\texttt{pointA}, \texttt{pointB}, 10000, 200)\}$$
$$\Pi^{\text{state}}_{\mathbf{heli2}}(\mathcal{BS}(\mathbf{heli1})) = \{\mathbf{in}(\texttt{pos}, \mathbf{heli2} : \mathit{getPos}())\}$$
$$\Pi^{\text{action}}_{[\mathbf{heli2},\mathbf{tank1}]}(\mathcal{BS}(\mathbf{heli1})) = \{\mathbf{P}\mathit{drive}(\texttt{pointX}, \texttt{pointY}, 40)\}$$
$$\Pi^{\text{state}}_{[\mathbf{heli2},\mathbf{tank1}]}(\mathcal{BS}(\mathbf{heli1})) = \{\mathbf{in}(\texttt{pos}, \mathbf{tank1} : \mathit{getPos}())\}$$

These sets formalize the following:

- $\Pi^{\text{action}}_{\texttt{heli2}}(\mathcal{B}S(\texttt{heli1}))$ *describes* **heli1** *'s beliefs about* **heli2** *'s actions and it says that it is possible to fly from pointA to pointB at 10000 feet and 200 knots.*
- $\Pi^{\text{state}}_{\texttt{heli2}}(\mathcal{B}S(\texttt{heli1}))$ *describes* **heli1** *'s beliefs about* **heli2** *'s state and it says that its position is* `pos`*.*
- $\Pi^{\text{action}}_{[\texttt{heli2},\texttt{tank1}]}(\mathcal{B}S(\texttt{heli1}))$ *describes* **heli1** *'s beliefs about* **heli2** *'s beliefs about* **tank1** *'s actions, and it says that it is possible to drive from pointX to pointY at 40 miles per hour.*
- $\Pi^{\text{state}}_{[\texttt{heli2},\texttt{tank1}]}(\mathcal{B}S(\texttt{heli1}))$ *describes* **heli1** *'s beliefs about* **heli2** *'s beliefs about* **tank1** *'s state, and it says that its position is* `pos`*.*

Obviously for $\mathfrak{a}$ to make a guess about agent $\mathbf{b}$'s behaviour, agent $\mathfrak{a}$ not only needs a belief table and a belief semantics table, but $\mathfrak{a}$ also needs to guess about $\mathbf{b}$'s action base, action program as well as the action and integrity constraints used by $\mathbf{b}$.

This is very much like having a guess about $\mathbf{b}$'s software package which we motivated and illustrated just before Definition 9.1 on page 407 (see the notion of *compatible* code call condition).

For notational convenience and better readability we merge all these ingredients into a set $\Gamma^{\mathfrak{a}}(\mathbf{b})$.

Definition 9.10 ($\Gamma^{\mathfrak{a}}(\mathfrak{b})$, $\mathsf{Info}(\mathfrak{a})$)
For agents $\mathfrak{a}, \mathfrak{b} \in \mathsf{A}$, *we denote by* $\Gamma^{\mathfrak{a}}(\mathfrak{b})$ *the following list of all beliefs that agent* $\mathfrak{a}$ *holds about another agent* $\mathfrak{b}$*: the software package* $\mathcal{S}^{\mathfrak{a}}(\mathfrak{b})$, *the* action base $\mathcal{AB}^{\mathfrak{a}}(\mathfrak{b})$, *the* action program $\mathcal{P}^{\mathfrak{a}}(\mathfrak{b})$, *the* integrity constraints $\mathcal{IC}^{\mathfrak{a}}(\mathfrak{b})$ *and the* action constraints $\mathcal{AC}^{\mathfrak{a}}(\mathfrak{b})$. $\Gamma^{\mathfrak{a}}(\mathfrak{b})$ *may also contain these objects for sequences* $\boldsymbol{\sigma} = [\mathfrak{b}, \mathfrak{c}]$ *instead of* $\mathfrak{b}$*: we use therefore also the notation* $\Gamma^{\mathfrak{a}}([\mathfrak{b}, \mathfrak{c}])$. $\Gamma^{\mathfrak{a}}(\boldsymbol{\sigma})$ *represents* $\mathfrak{a}$*'s beliefs about* $\mathfrak{b}$*'s beliefs about* $\mathfrak{c}$.

In addition, given an agent $\mathfrak{a}$, *we will often use the notation* $\mathsf{Info}(\mathfrak{a})$ *to denote the software package* $\mathcal{S}^{\mathfrak{a}}$, *the action base* $\mathcal{AB}$, *the action program* $\mathcal{P}$, *the integrity constraints* $\mathcal{IC}$ *and action constraints* $\mathcal{AC}$ *used by agent* $\mathfrak{a}$. *Thus we define* $\mathsf{Info}(\mathfrak{a}) =_{def} \Gamma^{\square}(\mathfrak{a})$.

The set $\Gamma^{\mathfrak{a}}(\mathfrak{b})$ is very important and therefore we introduce the corresponding software code calls, thereby extending our original package $\mathcal{S}$.

Definition 9.11 (Extended Code Calls, $\mathcal{S}^{\mathbf{ext}}$)

Given an agent $\mathfrak{a}$, we will from now on distinguish between basic *and* extended *code calls (resp. conditions). The basic code calls refer to the package $\mathcal{S}$, while the latter refer to the extended software package which also contains*

1. *the following function of the belief table:*
   (a) $\mathfrak{a}$: ***belief_table()***, *which returns the full belief table of agent $\mathfrak{a}$, as a set of triples $\langle \mathfrak{h}, \phi, \chi_{\mathcal{B}} \rangle$,*
2. *the following functions of the belief semantics table:*
   (b) $\mathfrak{a}$: ***belief_sem_table()***, *which returns the full belief semantics table, as a set of pairs $\langle \mathfrak{h}, \mathcal{B}Sem^{\mathfrak{a}}_{\mathfrak{h}} \rangle$,*
   (c) $\mathfrak{a}$: ***bel_semantics($\mathfrak{h}, \phi, \psi$)***, *which returns* **true** *when $\phi \models_{\mathcal{B}Sem^{\mathfrak{a}}_{\mathfrak{h}}} \psi$ and* **false** *otherwise.*

3. *the following functions, which implement for every sequence σ the beliefs of agent a about σ as described in $\Gamma^{a}(\sigma)$:*
   (d) a:***software_package(σ)***, *which returns the set* $\mathcal{S}^{a}(\sigma)$,
   (e) a:***action_base(σ)***, *which returns the set* $\mathcal{AB}^{a}(\sigma)$,
   (f) a:***action_program(σ)***, *which returns the set* $\mathcal{P}^{a}(\sigma)$,
   (g) a:***integrity_constraints(σ)***, *which returns the set* $\mathcal{IC}^{a}(\sigma)$
   (h) a:***action_constraints(σ)***, *which returns the set* $\mathcal{AC}^{a}(\sigma)$,

4. *the following functions which simulate the state of another agent* $\mathbf{b}$ *or a sequence* σ,

   (i) $\mathbf{a}$:***bel_ccc_act***(σ)*, which returns all the code call conditions and action status atoms that* $\mathbf{a}$ *believes are true in* σ*'s state. We write these objects in the form "***in*(,)*" (resp. "*$Op\,\alpha$*" for action status atoms) in order to distinguish them from those that have to be checked in* $\mathbf{a}$*'s state.*

   (j) $\mathbf{a}$:***not_bel_ccc_act***(σ)*, which returns all the code call conditions and action status atoms that* $\mathbf{a}$ *does not believe to be true in* σ*'s state.*

We also write $\mathcal{S}^{ext}$ *for this extended software package and distinguish it from the original* $\mathcal{S}$ *from which we started.*

9.4 Feasible Belief Status Sets

- Consider now an agent $\mathfrak{a}$ with associated structures, $\mathsf{Info}(\mathfrak{a})$.
- Suppose $\mathcal{BS}$ is an arbitrary status set. We would like to first identify the conditions that determine whether it "makes sense" for agent $\mathfrak{a}$ to hold the set of beliefs prescribed by $\mathcal{BS}$.
- Intuitively, $\mathcal{BS}$ is feasible *if and only if* it satisfies two types of conditions
  - conditions on the agent $\mathfrak{a}$, and
  - conditions on the beliefs of agent $\mathfrak{a}$ about other agents $\mathfrak{b}$ or sequences σ.

Conditions on agent a (as in the classical case):

1. **Deontic and action consistency:** $\mathcal{BS}$ must not contain any inconsistencies.
2. **Deontic and action closure:** This condition says that $\mathcal{BS}$ must be closed under the deontic operations.
3. **Closure under rules of $\mathcal{BP}$:** Furthermore, if we have a rule in $\mathcal{BP}$ having a ground instance whose body's code-call conditions are all true in the current agent state, and whose action status atoms and belief literals are true in $\mathcal{BS}$, then the head of that (ground) rule must be in $\mathcal{BS}$.
4. **State consistency:** Suppose we concurrently execute all actions in the set Todo. Then the new state that results must be consistent with the integrity constraints associated with agent a.

Conditions on beliefs of agent $\mathfrak{a}$ about other agents $\mathfrak{b}$ (new conditions):

5. **Local coherence:** This condition requires that for any agent $\mathfrak{b}$, every induced status set $\Pi_{\mathfrak{b}}^{\text{action}}(\mathcal{BS})$ is feasible (in the original sense) with respect to the induced state $\Pi_{\mathfrak{b}}^{\text{state}}(\mathcal{BS})$ and action program $\mathcal{P}^{\mathfrak{a}}(\mathfrak{b})$. Furthermore a similar condition must hold for any sequence σ instead of just $\mathfrak{b}$.
6. **Compatibility with BT$^{\mathfrak{a}}$:** We have to ensure that (1) all belief atoms of the basic belief table are contained in $\mathcal{BS}$ and that (2) whenever a belief condition is true, then the corresponding belief formula is true in $\mathcal{BS}$.
7. **Compatibility with BSemT$^{\mathfrak{a}}$:** If $\langle \mathfrak{b}, \mathcal{B}Sem_{\mathfrak{b}}^{\mathfrak{a}} \rangle$ is an entry in **BSemT$^{\mathfrak{a}}$**, we have to ensure that $\mathfrak{b}$'s induced state is closed under the semantics $\mathcal{B}Sem_{\mathfrak{b}}^{\mathfrak{a}}$.

9.5 Reducing map's to Ordinary Agent Programs

This can be done by

1. *transforming meta agent programs into agent programs*,
   (this is a source-to-source transformation: the belief atoms in a meta agent program are replaced by suitable code calls to the new datastructures),
2. *taking advantage of extended code calls* $\mathcal{S}^{ext}$ *as introduced in Definition 9.11 on page 449*.

- Suppose the belief table does not contain any belief conditions (i.e., it coincides with its basic belief table).
- Then if χ is any code call condition of agent $\mathbf{c}$, the extended code call atom

$$\mathbf{in}(\langle c, \chi, \mathbf{true}\rangle, a : \textit{belief\_table}())$$

  corresponds to the belief atom

$$\mathcal{B}_{\mathbf{a}}(\mathbf{c}, \chi).$$

- But beliefs can also be triggered by entries in the belief table and/or in the belief semantics table!

- What happens if the formula χ is not a code call, but again a belief formula, say $\mathcal{B}_{\mathbf{c}}(\mathbf{d}, \chi')$?
- Here is where the inductive definition of $\mathfrak{Trans}$ comes in. We map

$$\mathcal{B}_{\mathbf{a}}(\mathbf{c}, \mathcal{B}_{\mathbf{c}}(\mathbf{d}, \chi'))$$

  to

$$\mathbf{in}(\text{"}\chi'\text{"}, \mathfrak{a} : \textit{bel\_ccc\_act}([\mathrm{c}, \mathrm{d}])).$$

Our main theorem in this section states that there is indeed a uniform transformation $\mathfrak{Trans}$ from arbitrary meta agent programs (which can also contain nested beliefs) to agent programs such that the semantics are preserved:

$$\mathsf{Sem}(\mathcal{BP}) = \mathsf{Sem}(\mathfrak{Trans}(\mathcal{BP})) \tag{9.7}$$

where Sem is either the *feasible*, *rational* or *reasonable* belief status set semantics.

$$\begin{tikzcd}
\text{Compatible with Belief Semantics Belief Table} & \mathcal{BP} \arrow[r, "\mathfrak{Trans}"] & \mathcal{P} \\
& \mathcal{BS} \arrow[u, "\mathsf{Sem}^{\text{new}}"'] \arrow[r, "\mathfrak{Trans}"] \arrow[ur, phantom, "\mathcal{IC}^{\text{ext}}\ \text{Closure}"] & \mathcal{S} \arrow[u, "\mathsf{Sem}^{\text{old}}"']
\end{tikzcd} \tag{9.8}$$

Summary

This chapter was about an extension of agent programs by **beliefs**.

Agents have beliefs about other agents. How can we extend agent programs to incorporate such beliefs?

1. Belief Language
   (a) $\mathcal{B}_{\mathbf{a}}(\mathbf{b},\chi)$: $\mathbf{a}$ believes that in $\mathbf{b}$'s state the ccc χ holds.
   (b) Nestings are also allowed: $\mathcal{BL}^{\mathbf{a}}_{\infty}$.
2. Two new datastructures: **Belief-, Belief Semantics-Table**.
3. **Meta Agent programs**: allow beliefs in bodies of rules.

4. Semantics
   (a) **Belief Status sets $\mathcal{BS}$**: Status sets + Beliefs.
   (b) $\mathcal{BS}$'s induce: $\Pi_{\mathfrak{b}}^{\text{action}}(\mathcal{BS})$ and $\Pi_{\mathfrak{b}}^{\text{state}}(\mathcal{BS})$
   (c) Feasibility of $\mathcal{BS}$:
      i. Compatibility with Belief-, Belief Semantics-Table,
      ii. $\Pi_{\mathfrak{b}}^{\text{action}}(\mathcal{BS})$ must be feasible wrt. $\Pi_{\mathfrak{b}}^{\text{state}}(\mathcal{BS})$
5. Meta agent programs can be transformed to ordinary agent programs (using extended code calls) s.t. **Sem** $(\mathcal{BP})$=**Sem** $(\mathfrak{Trans}(\mathcal{BP}))$.

References

Apt, K., H. Blair, and A. Walker (1988). Towards a Theory of Declarative Knowledge. In J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, pp. 89–148. Washington DC: Morgan Kaufmann.

Arens, Y., C. Y. Chee, C.-N. Hsu, and C. Knoblock (1993). Retrieving and Integrating Data From Multiple Information Sources. *International Journal of Intelligent Cooperative Information Systems 2*(2), 127–158.

Arisha, K., F. Ozcan, R. Ross, V. S. Subrahmanian, T. Eiter, and S. Kraus (1999, March/April). IMPACT: A Platform for Collaborating Agents. *IEEE Intelligent Systems 14*, 64–72.

Bayardo, R., et al. (1997). Infosleuth: Agent-based Semantic Integration of Information in Open and Dynamic Environments. In J. Peckham (Ed.), *Proceedings of ACM SIGMOD Conference on Management of Data*, Tucson, Arizona, pp. 195–206.

Brink, A., S. Marcus, and V. Subrahmanian (1995). Heterogeneous Multimedia Reasoning. *IEEE Computer 28*(9), 33–39.

Chawathe, S., et al. (1994, October). The TSIMMIS Project: Integration of Heterogeneous Information Sources. In *Proceedings of the 10th Meeting of the Information Processing Society of Japan*, Tokyo, Japan. Also available via anonymous FTP from host db.stanford.edu, file /pub/chawathe/1994/tsimmis-overview.ps.

Dix, J., S. Kraus, and V. Subrahmanian (1999, September). Temporal agent programs. Technical Report CS-TR-4055, Dept. of CS, University of Maryland, College Park, MD 20752. currently under submission for a Journal.

Dix, J., M. Nanni, and V. S. Subrahmanian (2000). Probabilistic agent reasoning. *Transactions of Computational Logic 1*(2).

Dix, J., V. S. Subrahmanian, and G. Pick (2000). Meta Agent Programs. *Journal of Logic Programming 45*(1).

Eiter, T., V. Subrahmanian, and G. Pick (1999). Heterogeneous Active Agents, I: Semantics. *Artificial Intelligence 108*(1-2), 179–255.

Eiter, T., V. Subrahmanian, and T. J. Rogers (2000). Heterogeneous Active Agents, III: Polynomially Implementable Agents. *Artificial Intelligence 117*(1), 107–167.

Eiter, T. and V. S. Subrahmanian (1999). Heterogeneous Active Agents, II: Algorithms and Complexity. *Artificial Intelligence 108*(1-2), 257–307.

Genesereth, M. R. and S. P. Ketchpel (1994). Software Agents. *Communications of the ACM 37*(7), 49–53.

Rogers Jr., H. (1967). *Theory of Recursive Functions and Effective Computability*. New York: McGraw-Hill.

Subrahmanian, V., P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Özcan, and R. Ross (2000). *Heterogenous Active Agents*. MIT-Press.

Wiederhold, G. (1993). Intelligent Integration of Information. In *Proceedings of ACM SIGMOD Conference on Management of Data*, Washington, DC, pp. 434–437.

Wilder, F. (1993). *A Guide to the TCP/IP Protocol Suite*. Artech House.