IMPACT: A Platform for Heterogenous Agents

Lecture Course given at Ushuaia, Argentina

October 2000

Jürgen Dix,
University of Koblenz and Technical University of Vienna

1. *IMPACT* Architecture
2. The Code Call Mechanism
3. Actions and Agent Programs
4. Regular Agents
5. Meta Agent Reasoning
6. Probabilistic Agent Reasoning
7. Temporal Agent Reasoning

Based on the book

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

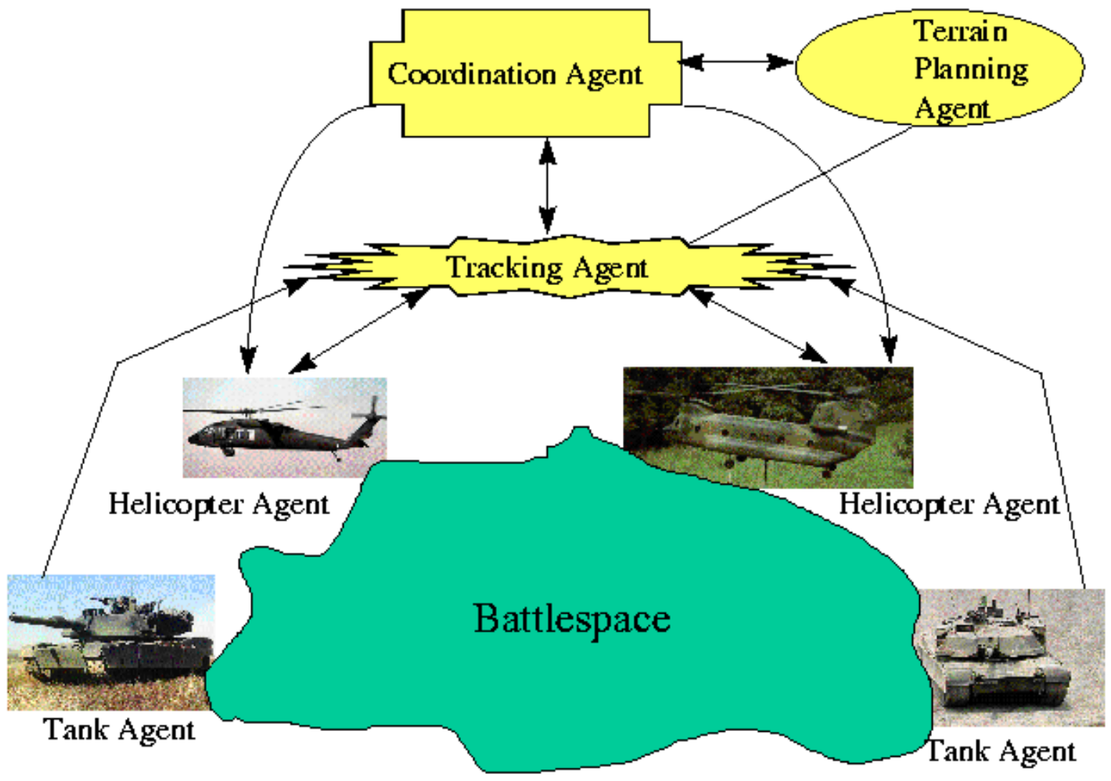
Timetable:

- 10 minutes to explain what is going on. Some sentences for each chapter.
- Chapter 1 can be entirely done in the remaining time.

Extensions of our Basic Approach

We did not yet incorporate the following important features, that are needed for most realistic applications:

- **Beliefs:** Agents hold beliefs about other agents.
  (Dix, Subrahmanian, and Pick 2000), Journal of Logic Programming, 46(1–2)*1–60.
- **Uncertainty:** Available information might be incomplete or uncertain.
  (Dix, Nanni, and Subrahmanian 2000), Transactions of Computational Logic, 1(2).
- **Time:** Agents use temporal reasoning. They make commitments to the future.
  (Dix, Kraus, and Subrahmanian 2001), Artificial Intelligence, to appear.


Coordination Agent
Terrain Planning Agent
Tracking Agent
Helicopter Agent
Helicopter Agent
Battlespace
Tank Agent
Tank Agent

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 1.2). 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. All three of **beliefs**, **time** and **uncertainty** enter here.

A coordination agent: coordination 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.

- **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*). 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.

5. Meta Agent Reasoning

5.1 Belief Language and Data Structures

5.2 Meta Agent Programs and Status Sets

5.3 Reducing map's to Ordinary Agent Programs

Timetable:

- Chapter 5 needs 30 minutes.

5 Meta Agent Programs

5.1 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.
- Most important are the beliefs about what holds in another agents state

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

- 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 $\mathcal{S}^{\mathfrak{b}}$.

Example 5.1 (Belief Atoms In CFIT*)

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

- $\mathcal{B}_{\texttt{heli1}}(\texttt{tank1}, \textbf{F}\textit{attack}(\texttt{pos1}, \texttt{pos2}))$
  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.

- $\mathcal{B}_{\texttt{heli3}}(\texttt{tank1}, \textbf{O}\textit{drive}(\texttt{pos1}, \texttt{pos2}, 35))$
  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 precise definition is very complicated!

A nested belief atom of the form

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

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

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

- We can use conjunctions with respect to different agents $\mathcal{B}_{\mathtt{a}}(\mathtt{b}, \chi) \wedge \mathcal{B}_{\mathtt{a}}(\mathtt{c}, \chi')$.
- We also use different nested levels of beliefs, like $\mathcal{B}_{\mathtt{a}}(\mathtt{b}, \chi) \wedge \mathcal{B}_{\mathtt{a}}(\mathtt{c}, \mathcal{B}_{\mathtt{c}}(\mathtt{d}, \chi'))$.

Example 5.2 (Belief Formulae for CFIT*)

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

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

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

- $\mathcal{B}_{\texttt{coord}}(\texttt{tank1}, \mathcal{B}_{\texttt{tank1}}(\texttt{heli1}, \mathcal{B}_{\texttt{heli1}}(\texttt{tank2}, \texttt{in}(\texttt{pos2}, \texttt{tank2}:\textit{getPosition()}))))$.
  This formula is in $\mathcal{BL}_3^{\texttt{coord}}$. *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:

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

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

Example 5.3 (Basic Belief Table for CFIT* Agents)
We define suitable basic belief tables for agent tank1.

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

Table 5.1: A Basic Belief Table for agent tank1.

What kind of operations should we support on belief tables?

We distinguish between two different types:

1. For a given agent $\mathtt{h}$, other than $\mathtt{a}$, we may want to select all entries in the table having $\mathtt{h}$ 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.

5.1.1 Belief Semantics Table

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

Example 5.4 (Belief Semantics Tables for CFIT* Agents)

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

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

$$\mathcal{B}_{\texttt{tank1}}(\texttt{tank1.1}, \chi) \rightarrow \chi.$$

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

2. $\mathcal{BSem}^{\texttt{tank1}}_{\texttt{heli2}}$: *The smallest entailment relation satisfying the schema*

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

   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 $\mathcal{BL}_i^{\mathbf{h}} \times \mathcal{BL}_i^{\mathbf{h}}$.

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

$$(1) \quad \mathcal{B}_{\mathbf{h}_2}(\mathbf{h}, \chi) \Rightarrow \mathcal{B}_{\mathbf{h}_2}(\mathbf{h}', \chi)$$

$$(2) \quad \mathcal{B}_{\mathbf{h}_2}(\mathbf{h}, \chi) \Rightarrow \chi$$

The first axiom refers to different agents $\mathbf{h}, \mathbf{h}'$ while the second combines different *levels* of belief atoms. In many applications, however, such axioms will not occur: $\mathbf{h} = \mathbf{h}'$ is fixed and the axioms operate on the same level i of belief formulae.

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

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

This is indeed possible.

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

5.2 Meta Agent Programs and Status Sets

Definition 5.1 (Meta Agent Program (map) $\mathcal{BP}$)
A meta agent rule, *(mar for short), for agent* $\mathfrak{a}$ *is an expression* r *of the form*

$$Op\,\alpha(\vec{t}) \leftarrow L_1, \ldots, L_n \tag{5.4}$$

where $Op\,\alpha(\vec{t})$ *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* $\mathcal{BLit}_\infty(\mathfrak{a}, \mathbb{A})$.

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 5.5 (map's For CFIT*-Agents)
Let heli1 *'s meta agent program be as follows:*

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

where $\mathit{attack}(\mathtt{P1},\mathtt{P2})$ *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:*

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

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 5.2 (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 $\mathcal{BA}\mathrm{t}_{\infty}(\mathfrak{a},\mathcal{A})$ 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 3.15 on page 179).

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

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

- In contrast to agent programs without beliefs we now have **to cope with all agents about which $\mathfrak{a}$ 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 $\mathbf{BT}^{\mathfrak{a}}$ may contain nested beliefs and, by the belief semantics table $\mathbf{BSemT}^{\mathfrak{a}}$, such nested beliefs may imply (trigger) other beliefs.

5.3 Reducing map's to Ordinary Agent Programs

Definition 5.3 (Extended Code Calls, $\mathcal{S}^{\mathbf{ext}}$)
Given an agent $\mathtt{a}$*, 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) $\mathtt{a}$:***belief_table()***, *which returns the full belief table of agent* $\mathtt{a}$*, as a set of triples* $\langle \mathtt{h}, \phi, \chi_{\mathcal{B}} \rangle$,
2. *the following functions of the belief semantics table:*
   (b) $\mathtt{a}$:***belief_sem_table()***, *which returns the full belief semantics table, as a set of pairs* $\langle \mathtt{h}, \mathcal{B}Sem^{\mathtt{a}}_{\mathtt{h}} \rangle$,
   (c) $\mathtt{a}$:***bel_semantics(***$\mathtt{h}, \phi, \psi$***)***, *which returns* **true** *when* $\phi \models_{\mathcal{B}Sem^{\mathtt{a}}_{\mathtt{h}}} \psi$ *and* **false** *otherwise.*

3. *the following functions, which implement for every sequence* σ *the beliefs of agent* $\mathfrak{a}$ *about* σ *as described in* $\Gamma^{\mathfrak{a}}(\sigma)$*:*

   (d) $\mathfrak{a}$:***software_package(σ)***, *which returns the set* $\mathcal{S}^{\mathfrak{a}}(\sigma)$,

   (e) $\mathfrak{a}$:***action_base(σ)***, *which returns the set* $\mathcal{AB}^{\mathfrak{a}}(\sigma)$,

   (f) $\mathfrak{a}$:***action_program(σ)***, *which returns the set* $\mathcal{P}^{\mathfrak{a}}(\sigma)$,

   (g) $\mathfrak{a}$:***integrity_constraints(σ)***, *which returns the set* $\mathcal{IC}^{\mathfrak{a}}(\sigma)$

   (h) $\mathfrak{a}$:***action_constraints(σ)***, *which returns the set* $\mathcal{AC}^{\mathfrak{a}}(\sigma)$,

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

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

   (j) $\mathfrak{a}$:***not_bel_ccc_act(σ)***, *which returns all the code call conditions and action status atoms that* $\mathfrak{a}$ *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.*

We now

1. *transform 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. *take advantage of extended code calls* $\mathcal{S}^{ext}$.

- 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

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

  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 $\mathcal{B}_{\mathfrak{c}}(\mathfrak{d}, \chi')$?
- Here is where the inductive definition of $\mathfrak{Trans}$ comes in. We map

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

to

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

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:

$$\mathbf{Sem}(\mathcal{BP}) = \mathbf{Sem}(\mathfrak{Trans}(\mathcal{BP})) \tag{5.5}$$

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

$$\begin{array}{lcl} \mathcal{BP} & \xrightarrow{\mathfrak{Trans}} & \mathcal{P} \\ \text{Compatible with} & & \\ \text{Belief Semantics} & & \mathit{IC}^{\text{ext}} \\ \text{Belief Table} \uparrow \mathbf{Sem}^{\text{new}} & & \text{Closure} \uparrow \mathbf{Sem}^{\text{old}} \\ \mathcal{BS} & \xrightarrow{\mathfrak{Trans}} & S \end{array} \tag{5.6}$$

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 (2001). Temporal agent reasoning. *Artificial Intelligence to appear*.

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 46*(1-2), 1–60.

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.