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.

6. Probabilistic Agent Reasoning

Overview

6.1 Probabilistic Code Calls

6.2 Probabilistic Agent Programs

6.3 Kripke Style Semantics

Timetable:

- Chapter 6 needs 30 minutes.

6 Probabilistic Agent Reasoning

6.1 Probabilistic Code Calls

Imagine a surveillance example, where *surv*:*identify*(`image1`) tries to identify all objects in a given image: it is well known that this is an uncertain task.

Some objects may be identified with 100% certainty, while in other cases, it may only be possible to say it is either a T-72 tank with 40–50% probability.

Definition 6.1 (Random Variable of Type τ)
A random variable *of type* τ *is a finite set* **RV** *of objects of type* τ*, together with a probability distribution* $\wp$ *that assigns real numbers in the unit interval* $[0,1]$ *to members of* **RV** *such that* $\Sigma_{o \in \mathbf{RV}} \wp(o) \leq 1$.

Uncertainty can be captured as follows.

Definition 6.2 (Probabilistic Code Call $a :_{RV} f(d_1, \ldots, d_n)$)

Suppose $a : f(d_1, \ldots, d_n)$ *is a code call whose output type is* τ. *The* probabilistic code call *associated with* $a : f(d_1, \ldots, d_n)$, *denoted* $a :_{RV} f(d_1, \ldots, d_n)$, *returns a set of random variables of type* τ *when executed.*

Example 6.1

Consider the code call $\mathit{surv}:_{\mathbf{RV}} \mathit{identify}(\texttt{image1})$. *This code call may return the following two random variables.*

$\langle\{t72, t80\}, \{\langle t72, 0.5\rangle, \langle t80, 0.4\rangle\}\rangle$ and $\langle\{t60, t84\}, \{\langle t60, 0.3\rangle, \langle t84, 0.7\rangle\}\rangle$

This says that the image processing algorithm has identified two objects in image1:

- *The first object is either a T-72 or a T-80 tank with 50% and 40% probability, respectively, while*
- *the second object is either a T-60 or a T-84 tank with 30% and 70% probability respectively.*

Probabilistic cc's and ccc's look exactly like ordinary cc's and ccc's—however, as a probabilistic code call returns a set of *random variables*, **probabilistic code call atoms are true or false with some probability.**

Example 6.2

Consider the probabilistic code call condition

$$\mathbf{in}(\mathtt{X}, \mathbf{surv} :_{\mathbf{RV}} \mathit{identify}(\mathtt{image1})) \,\&\, \mathbf{in}(a1, \mathbf{surv} :_{\mathbf{RV}} \mathit{turret}(\mathtt{X})).$$

This ccc attempts to find all vehicles in "image1" with a gun turret of type a1. Let us suppose that the first cc is as on the previous page, but gives back only the first random variable.

When this result (X) is passed to the second code call, it returns one random variable with two values—a1 with probability 30% and a2 with probability 65%.

What is the probability that the code call condition above is satisfied by a particular assignment to X?

Let's suppose X is assigned T72. If all T72's have a2-type turrets, then the answer is "0".

Let's suppose X is assigned T80. If the vehicule and turret identification is independent, then the answer is "$0.4 \times 0.3 = 0.12$*".*

Example 6.3

Suppose we consider a code call cc returning the following two random variables.

$$\begin{aligned} \mathbf{RV}_1 &= \langle \{a,b\}, \wp_1 \rangle \\ \mathbf{RV}_2 &= \langle \{b,c\}, \wp_2 \rangle \end{aligned}$$

Suppose $\wp_1(a) = 0.9, \wp_1(b) = 0.1, \wp_2(b) = 0.8, \wp_2(c) = 0.1.$

What is the probability that b is in the result of the code call cc?

Answering this question is problematic.

Definition 6.3 (Probabilistic State of an Agent)
The probabilistic state of an agent $\mathbf{a}$ *at any given point* t *in time, denoted* $O^p(t)$*, consists of the set of all instantiated data objects and random variables of types contained in* $\mathcal{T}_{\mathbf{a}}$*.*

Definition 6.4 (Satisfying a Code Call Atom)
Suppose $\mathbf{a}:_{\mathbf{RV}}\mathbf{f}(\mathtt{d}_1,\ldots,\mathtt{d}_n)$ *is a ground probabilistic code call and* o *is an object of the output type of this code call w.r.t. probabilistic agent state* O^p*. Suppose* $[\ell,u]$ *is a closed, nonempty subinterval of the unit interval* $[0,1]$*.*

- $o \models^{[\ell,u]}_{O^p} \mathbf{in}(\mathtt{X},\mathbf{a}:_{\mathbf{RV}}\mathbf{f}(\mathtt{d}_1,\ldots,\mathtt{d}_n))$
  if there is a $(Y,\wp)$ *in the answer returned by evaluating* $\mathbf{a}:_{\mathbf{RV}}\mathbf{f}(\mathtt{d}_1,\ldots,\mathtt{d}_n)$ *w.r.t.* O^p *such that* $o \in Y$ *and* $\ell \leq \wp(o) \leq u$*.*
- $o \models^{[\ell,u]}_{O^p} \mathbf{not\_in}(\mathtt{X},\mathbf{a}:_{\mathbf{RV}}\mathbf{f}(\mathtt{d}_1,\ldots,\mathtt{d}_n))$
  if for all random variables $(Y,\wp)$ *returned by evaluating* $\mathbf{a}:_{\mathbf{RV}}\mathbf{f}(\mathtt{d}_1,\ldots,\mathtt{d}_n)$ *w.r.t.* O^p*, either* $o \notin Y$ *or* $\wp(o) \notin [\ell,u]$*.*

Probabilistic code call conditions are defined in exactly the same way as code call conditions. However, extending the above definition of “satisfaction” to probabilistic code call conditions is highly problematic because (as shown in Examples 6.2, 6.3)

the probability that a conjunction is true depends not only on the probabilities of the individual conjuncts, but also on the dependencies between the events denoted by these conjuncts.

We allow the user to specify certain strategies.

Definition 6.5 (Probabilistic Conjunction Strategy $\otimes$)

A probabilistic conjunction strategy *is a mapping* $\otimes$ *which maps a pair of probability intervals to a single probability interval satisfying the following axioms:*

1. ***Bottomline:*** $[L_1,U_1] \otimes [L_2,U_2] \leq [\min(L_1,L_2), \min(U_1,U_2)]$ *where* $[x,y] \leq [x',y']$ *if* $x \leq x'$ *and* $y \leq y'$.
2. ***Ignorance:*** $[L_1,U_1] \otimes [L_2,U_2] \subseteq [\max(0,L_1+L_2-1), \min(U_1,U_2)]$.
3. ***Identity:*** *When* $(e_1 \wedge e_2)$ *is consistent and* $[L_2,U_2] = [1,1]$, $[L_1,U_1] \otimes [L_2,U_2] = [L_1,U_1]$.
4. ***Annihilator:*** $[L_1,U_1] \otimes [0,0] = [0,0]$.
5. ***Commutativity:*** $[L_1,U_1] \otimes [L_2,U_2] = [L_2,U_2] \otimes [L_1,U_1]$.
6. ***Associativity:*** $([L_1,U_1] \otimes [L_2,U_2]) \otimes [L_3,U_3] = [L_1,U_1] \otimes ([L_2,U_2] \otimes [L_3,U_3])$.
7. ***Monotonicity:*** $[L_1,U_1] \otimes [L_2,U_2] \leq [L_1,U_1] \otimes [L_3,U_3]$ *if* $[L_2,U_2] \leq [L_3,U_3]$.

The concept of a conjunction strategy is very general, and has as special cases, the following well known ways of combining probabilities.

1. When we do not know the dependencies between e_1, e_2, we may use the conjunction strategy $\otimes_{\mathbf{ig}}$ defined as $([L_1,U_1]\otimes_{\mathbf{ig}}[L_2,U_2]) \equiv [\max(0,L_1+L_2-1),\min(U_1,U_2)]$.
2. When e_1, e_2 have maximal overlap, use the positive correlation conjunctive strategy $\otimes_{\mathbf{pc}}$ defined as $([L_1,U_1]\otimes_{\mathbf{pc}}[L_2,U_2]) \equiv [\min(L_1,L_2),\min(U_1,U_2)]$.
3. When e_1, e_2 have minimal overlap, use the negative correlation conjunctive strategy $\otimes_{\mathbf{nc}}$ defined as $([L_1,U_1]\otimes_{\mathbf{nc}}[L_2,U_2]) \equiv [\max(0,L_1+L_2-1),\max(0,U_1+U_2-1)]$.
4. When the two events occur independently, use the independence conjunction strategy $([L_1,U_1]\otimes_{\mathbf{in}}[L_2,U_2]) = [L_1 \cdot L_2, U_1 \cdot U_2]$.

6.2 Probabilistic Agent Programs

We assume the existence of an *annotation language* $\mathbf{L}^{ann}$—the constant symbols of $\mathbf{L}^{ann}$ are the real numbers in the unit interval $[0,1]$.

Definition 6.6 (Annotation Item)
We define annotation items inductively:

- *Every constant and every variable of* $\mathbf{L}^{ann}$ *is an annotation item.*
- *If* f *is an annotation function of arity* n *and* $\mathsf{ai}_1,\ldots,\mathsf{ai}_n$ *are annotation items, then the term* $f(\mathsf{ai}_1,\ldots,\mathsf{ai}_n)$ *is an annotation item.*

An annotation item is ground *if no annotation variables occur in it.*

Definition 6.7 (Annotation $[\mathsf{ai}_1, \mathsf{ai}_2]$)
If $\mathsf{ai}_1, \mathsf{ai}_2$ *are annotation items, then the term* $[\mathsf{ai}_1, \mathsf{ai}_2]$ *is an* annotation. *If* $\mathsf{ai}_1, \mathsf{ai}_2$ *are both ground, then* $[\mathsf{ai}_1, \mathsf{ai}_2]$ *is a* ground annotation.

For instance, $[0, 0.4]$, $[0.7, 0.9]$, $[0.1, \frac{\mathtt{V}}{2}]$, $[\frac{\mathtt{V}}{4}, \frac{\mathtt{V}}{2}]$ are all annotations. The annotation $[0.1, \frac{\mathtt{V}}{2}]$ denotes an interval only when a value in $[0, 1]$ is assigned to the variable $\mathtt{V}$.

Definition 6.8 (Annotated Code Call Condition $\chi : \langle [ai_1, ai_2], \otimes \rangle$)
If χ is a probabilistic code call condition, $\otimes$ is a conjunction strategy, and $[ai_1, ai_2]$ is an annotation, then $\chi : \langle [ai_1, ai_2], \otimes \rangle$ *is an* annotated code call condition.
$\chi : \langle [ai_1, ai_2], \otimes \rangle$ *is* ground *if there are no variables in either χ or in $[ai_1, ai_2]$.*

For example, when X is ground,

$$\mathbf{in}(\mathtt{X}, \mathit{surv} :_{\mathbf{RV}} \mathit{identify}(\mathtt{image1})) \,\&\, \mathbf{in}(\mathtt{a1}, \mathit{surv} :_{\mathbf{RV}} \mathit{turret}(\mathtt{X})) : \langle [0.3, 0.5], \otimes_{\mathbf{ig}} \rangle$$

is true *if and only if* the probability that X is identified by the surv agent and that the turret is identified as being of type a1 lies between 30 and 50% assuming that nothing is known about the dependencies between turret identifications and identifications of objects by surv.

Definition 6.9 (Probabilistic Agent Programs $\mathcal{PP}$)
Suppose Γ *is an annotated code call condition, and* $A, L_1, \ldots, L_n$ *are status atoms. Then*

$$A \leftarrow \Gamma \,\&\, L_1 \,\&\, \ldots \,\&\, L_n \tag{6.7}$$

is a probabilistic action rule.

A probabilistic agent program *(pap for short) is a finite set of probabilistic action rules.*

It is important to note in the above definition that in a probabilistic action rule, status atoms are *not* annotated—uncertainty is present only in the state, and on the basis of this uncertainty, the agent must determine what it is obliged to do, forbidden from doing, etc.

$$
\begin{aligned}
\mathbf{Do}\, send\_warn(\mathtt{X}) \quad \leftarrow \quad & \mathbf{in}(\mathtt{F}, surv : file(\mathtt{imagedb})) \,\&\\
& \mathbf{in}(\mathtt{X}, surv :_{\mathbf{RV}} identify(\mathtt{F})) \,\&\\
& \mathbf{in}(\mathtt{a1}, surv :_{\mathbf{RV}} turret(\mathtt{X}))) : \langle [0.7, 1.0], \otimes_{ig} \rangle\\
& \neg \mathbf{F} send\_warn(\mathtt{X}).\\
\mathbf{F} send\_warn(\mathtt{X}) \quad \leftarrow \quad & \mathbf{in}(\mathtt{F}, surv : file(\mathtt{imagedb})) \,\&\\
& \mathbf{in}(\mathtt{X}, surv :_{\mathbf{RV}} identify(\mathtt{F})) \,\&\\
& \mathbf{in}(\mathtt{L}, geo :_{\mathbf{RV}} getplnode(\mathtt{X.location})) \,\&\\
& \mathbf{in}(\mathtt{L}, geo :_{\mathbf{RV}} range(\mathtt{100}, \mathtt{100}, \mathtt{20})).
\end{aligned}
$$

Definition 6.10 (Feasible Probabilistic Status Set)
Suppose $\mathcal{PP}$ is an agent program and O^p is a probabilistic agent state. A probabilistic status set $\mathcal{PS}$ is feasible *for $\mathcal{PP}$ on O^p if the following conditions hold:*

($\mathcal{PS}$1): $\mathbf{App}_{\mathcal{PP},O_S}(\mathcal{PS}) \subseteq \mathcal{PS}$ (closure under the program rules);

($\mathcal{PS}$2): *$\mathcal{PS}$ is deontically and action consistent* (deontic/action consistency);

($\mathcal{PS}$3): *$\mathcal{PS}$ is action closed and deontically closed* (deontic/action closure);

($\mathcal{PS}$4): *$\mathcal{PS}$ is state consistent* (state consistency).

Definition 6.11 (Deontic and Action Consistency)
A probabilistic status set $\mathcal{PS}$ is deontically consistent *with respect to a probabilistic agent state O^p if, by definition, it satisfies the following rules for any ground action α:*

- *If* $\mathbf{O}\alpha \in \mathcal{PS}$, *then* $\mathbf{W}\alpha \notin \mathcal{PS}$.
- *If* $\mathbf{P}\alpha \in \mathcal{PS}$, *then* $\mathbf{F}\alpha \notin \mathcal{PS}$.
- *If* $\mathbf{P}\alpha \in \mathcal{PS}$, *then* $O^p \models^{[1,1]} Pre(\alpha)$.

A probabilistic status set $\mathcal{PS}$ is action consistent *w.r.t. O^p if, by definition, for every action constraint of the form*

$$\{\alpha_1(\vec{X}_1), \ldots, \alpha_k(\vec{X}_k)\} \hookleftarrow \chi \tag{6.8}$$

either $O^p \not\models^{[1,1]} \chi$ *or* $\{\alpha_1(\vec{X}_1), \ldots, \alpha_k(\vec{X}_k)\} \not\subseteq \mathbf{Do}(\mathcal{PS})$.

Definition 6.12

Let $\mathcal{PP}$ be a probabilistic agent program, $\mathcal{PS}$ a probabilistic status set and O^p a probabilistic agent state. Assume further that each random variable contains exactly one object with probability 1. Then we can define the following mappings:

$\mathbf{Red}_1(\cdot)$, *which maps every random variable of the form $\langle\{o_{\mathbf{RV}}\},1\rangle$ to o:*

$$Red_1(\langle\{o_{\mathbf{RV}}\},1\rangle) = o.$$

$\mathbf{Red}_2(\cdot)$, *which maps annotated code call conditions to code call conditions by simply removing the annotations and the conjunction strategy:*

$$Red_2(\chi : \langle[\mathsf{ai}_1,\mathsf{ai}_2],\otimes\rangle) = \chi.$$

We can easily extend $Red_2(\cdot)$ to a mapping from arbitrary conjunctions of annotated code calls to conjunctions of code calls.

Red$_3(\cdot)$**,** *which maps every probabilistic agent program to a non-probabilistic agent program: it clearly suffices to define* $Red_3(\cdot)$ *on probabilistic agent rules. This is done as follows*

$$Red_3(A \leftarrow \Gamma \,\&\, L_1 \,\&\, \ldots \,\&\, L_n) = A \leftarrow Red_2(\Gamma) \,\&\, \&\, L_1 \,\&\, \ldots \,\&\, L_n.$$

Theorem 6.1 (Semantics of Agent Programs as an Instance of paps)
Suppose all random variables have the form

$$\langle \{object_{\mathbf{RV}}\}, 1 \rangle.$$

Then: ($\chi : \langle [\mathsf{ai}_1, \mathsf{ai}_2], \otimes \rangle$ *is a ground annotated ccc,* O^p *a probabilistic agent state)*

Satisfaction: *the satisfaction relations coincide, i.e.*

$$O^p \models^{[\mathsf{ai}_1,\mathsf{ai}_2]} \chi : \langle [\mathsf{ai}_1, \mathsf{ai}_2], \otimes \rangle \text{ if and only if } O^p \models Red_2(\chi : \langle [\mathsf{ai}_1, \mathsf{ai}_2], \otimes \rangle).$$

App-Operators: *the App-Operators coincide, i.e.*

$$\mathbf{App}_{Red_3(\mathcal{PP}),O^p}(\mathcal{PS}) = \mathbf{App}_{\mathcal{PP},O^p}(\mathcal{PS}).$$

Feasibility: *Feasible probabilistic status sets coincide with feasible status sets under our reductions, i.e.* $\mathcal{PS}$ *is a feasible probabilistic status set w.r.t.* $\mathcal{PP}$ *if and only if* $\mathcal{PS}$ *is a feasible status set w.r.t.* $Red_3(\mathcal{PP})$.

6.3 Kripke Style Semantics

Up to now, we assumed:

- An action can be executed only if its precondition is believed by the agent to be true in the agent state **with probability** 1.
- Every action that is permitted must also have a precondition that is believed to be true **with probability** 1.

Every probabilistic state implicitly determines a set of (ordinary) states that are "compatible" with it.

Definition 6.13 (Compatibility w.r.t. a Probabilistic State: $\mathsf{COS}(O^p)$)

Let O^p *be a probabilistic agent state. An (ordinary) agent state* O *is said to be* compatible *with* O^p *if, by definition, for every ground code call* $\mathbf{a}:f(\mathtt{d}_1,\ldots,\mathtt{d}_n)$*, it is the case that for every object* $o \in \mathbf{eval}(\mathbf{a}:f(\mathtt{d}_1,\ldots,\mathtt{d}_n), O)$*, there exists a random variable* $(X,\wp) \in \mathbf{eval}(\mathbf{a}:_{\mathbf{RV}} f(\mathtt{d}_1,\ldots,\mathtt{d}_n), O^p)$ *such that* $o \in X$ *and* $\wp(o) > 0$*, and there is no other object* $o' \in X$ *such that* $o' \in \mathbf{eval}(\mathbf{a}:f(\mathtt{d}_1,\ldots,\mathtt{d}_n), O)$*.*

We use the notation $\mathsf{COS}(O^p)$*.*

Example 6.4

Consider a probabilistic agent state O^p *with only two code calls* surv:*identify(*`image1`*)* *and* surv:*location(*`image1`*)*, *which respectively return the random variables*

$$\langle\{t80, t72, t70\}, \{\langle t80, 0.3\rangle, \langle t72, 0.7\rangle, \langle t70, 0.0\rangle\}\rangle$$

and $\langle\{loc2\}, \{\langle loc2, 0.8\rangle\}\rangle$. *The agent states compatible w.r.t.* O^p *are described in the following table:*

State	*Vehicle*	*Location*
1	*none*	*none*
2	*t80*	*none*
3	*t72*	*none*

State	*Vehicle*	*Location*
4	*none*	*loc2*
5	*t80*	*loc2*
6	*t72*	*loc2*

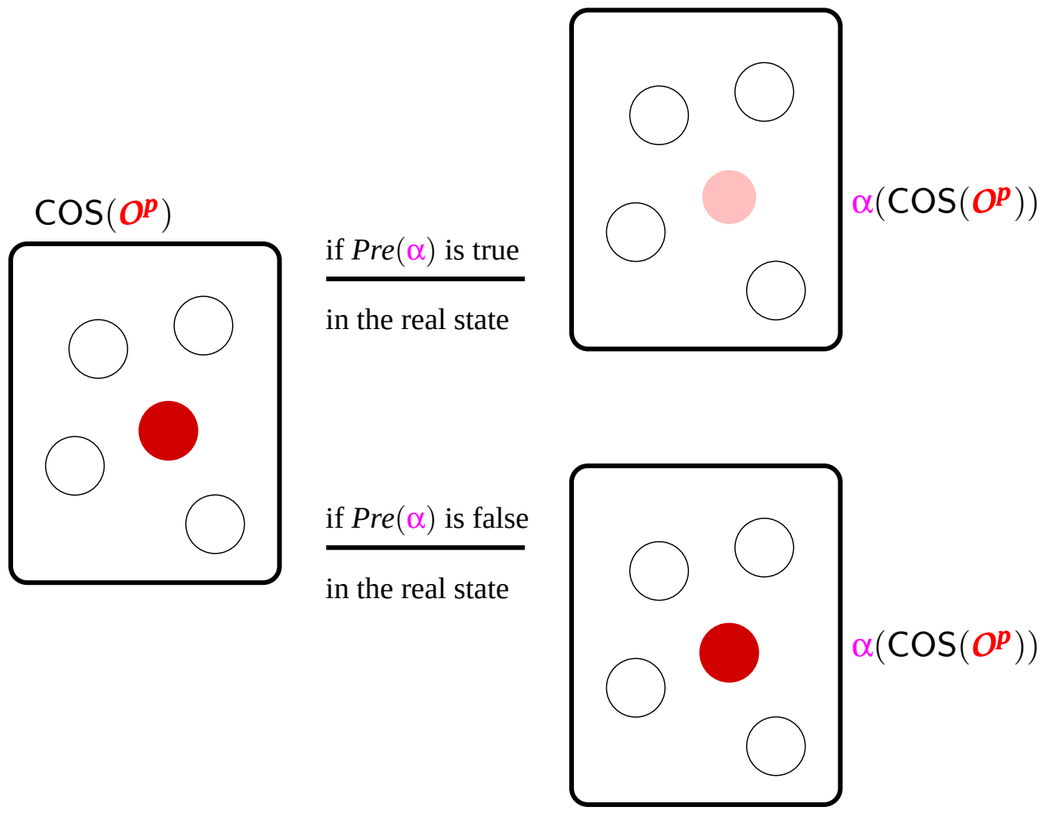

Figure 6.1: Applying an action.

It would be nice if

- agents were able to reason about the effects of their actions even when they are not exactly sure what the world state is.
  $\leadsto$ **Probabilistic Kripke Structures**
- actions could be applied even when the precondition is only true wrt. a certain probability $p < 1$.
  $\leadsto$ **p-Feasible Status Sets**

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.