Interactive Maryland Platform for Agents Collaborating Together

J. Dix, Th. Eiter

Universities of Maryland, Koblenz, TU Vienna

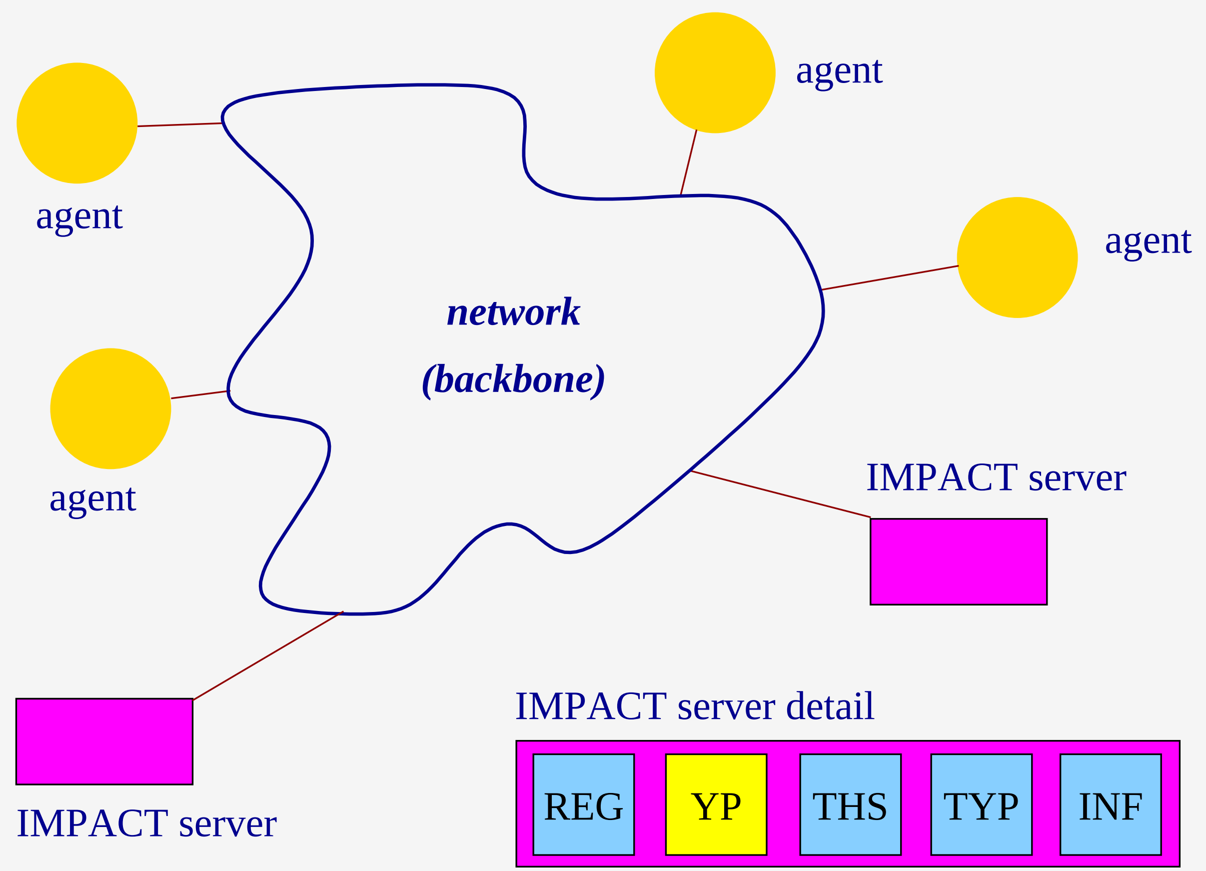
Part I. Basics

- System Outline
- Impact Agents
- Decision Component: Agent Programs

IMPACT goals

- A system for **creating** and **deploying** software agents.
- Custom-designed agents, or **"agentize" legacy code**.
- Build agents **on top** of a variety **of data structures**.
- Agent should take actions based on a **clearly articulated decision policy**.
- Incorporate aspects such as meta-knowledge, time, uncertainty, security.
- Provide tools for developing agents: **IADE** (*IMPACT* Agent Development Environment).

Overall IMPACT Architecture



IMPACT Servers

- **Registration**: creator enters agent to the system, which services it provides, and who may use them.
- **Yellow Pages**: Process requests from agents to find other agents providing some service (matchmaking).
- **Thesaurus**: requested when agent services are entered, or YP server performs matchmaking.
- **Types**: Server maintains a set of class hierarchies used by different agents and the inclusion relationship(s).

IMPACT Servers have been built in C (backend), with Java as well as *Tcl/Tk* frontends.

IMPACT Agents: Architecture

IMPACT Architecture Components

Message Box: Agents communicate through messages. *IMPACT* provides code to manage incoming/outgoing messages.

Action Policy: Every agent acts in accordance with certain principles articulated by the agent designer. This component is responsible for the agent decision making.

Legacy Data: This software (written in any language) manages a body of legacy data types. It is accessible through API **function calls**.

Agent State: At any time instant, the state of the agent is the set of objects the agent is currently managing.

Agent Programs

Specify action policy using a declarative language:

- **Code calls**: access software $\mathbf{in}(\mathtt{X}, \mathcal{S} : f(\mathtt{args}))$.
- **Action atoms**: $move(\mathtt{X1}, \mathtt{Y1}, \mathtt{X2}, \mathtt{Y2})$ STRIPS-like actions
- **Action status**: deontic modality $\mathbf{P}$ (permitted), $\mathbf{F}$ (forbidden), $\mathbf{O}$ (obliged), $\mathbf{W}$ (waived), $\mathbf{Do}$ (execute).

An **Agent Program** $\mathcal{P}$ is a finite collection of action rules

Example: Select driving lane (left or right)

$$\begin{aligned}
\mathbf{O}(go_rightmost) &\leftarrow \\
\mathbf{O}(drive(\mathtt{r_lane})) &\leftarrow \mathbf{Do}\,(go_rightmost), \mathbf{in}(\mathtt{r_lane}, \mathrm{status} : free_lanes()) \\
\mathbf{F}(drive(\mathtt{L})) &\leftarrow \mathbf{not_in}(\mathtt{L}, \mathrm{status} : free_lanes()) \\
\mathbf{Do}\,(drive(\mathtt{l_lane})) &\leftarrow \mathbf{F}(drive(\mathtt{r_lane})), \mathbf{in}(\mathtt{l_lane}, \mathrm{status} : free_lanes())
\end{aligned}$$

Semantics: Status Sets

A **status set** S is a collection of ground action status atoms $Op\,\alpha$. Status set S is **feasible** on an agent state, if

1. S is closed under rules of $\mathcal{P}$,
2. S satisfies deontic and action consistency,
3. S is deontically and action closed,
4. executing $\mathbf{Do}\,(\mathsf{S}) = \{\alpha \mid \mathbf{Do}\,(\alpha) \in \mathsf{S}\}$ leads to a consistent new state.

Example (both lanes free):

$$\mathsf{S} = \{\mathbf{O}(\mathtt{go\_rightmost}),\ \mathbf{Do}\,(\mathtt{go\_rightmost}),\ \mathbf{P}(\mathtt{go\_rightmost}),\ \mathbf{Do}\,(\mathit{drive}(\mathtt{r\_lane})),\ \mathbf{P}(\mathit{drive}(\mathtt{r\_lane})),\ \ldots\ \}.$$

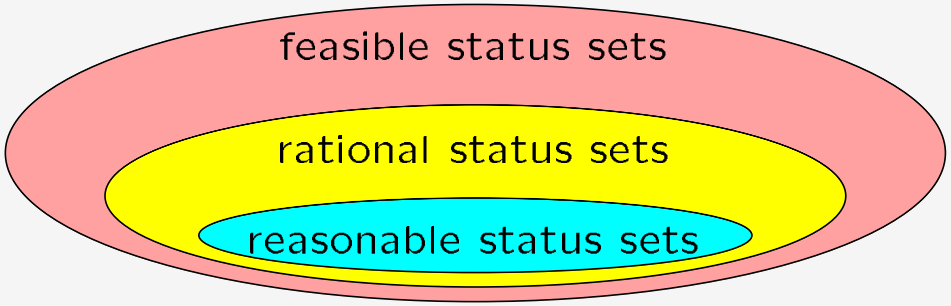
Item (4) requests a **concurrent action execution strategy**.

More Appealing Semantics

Feasible status set S may include atom A "unfoundedly".

Principle: Each atom A in S must be founded through a rule from $\mathcal{P}$, or deontic closure rules.

Use ideas from declarative logic programming



Rational Status Sets: intuitively, minimal (wrt $\subseteq$) FSS

Reasonable Status Sets: stable negation on top of rational status sets. Enables default rules

$$\mathbf{Do}\,\alpha \leftarrow \neg\mathbf{F}\alpha.$$

Part II. Extensions

- Meta Knowledge
- Temporal Aspects
- Uncertainty
- Security

Agent's beliefs about other Agents

- **Beliefs about agents states**: Belief atoms $\mathcal{B}_a(b, \chi)$:
  $\mathcal{B}_{\mathtt{Heli1}}(\mathrm{Tank1}, \mathsf{in}(\mathtt{pos1}, \mathrm{Tank1} : \mathit{GetPosition}(\mathtt{now})))$,
  $\mathcal{B}_{\mathtt{Heli3}}(\mathrm{Tank1}, \mathbf{O}\,\mathit{Drive}(\mathtt{pos1}, \mathtt{pos2}, 35))$.
  Iterate this: $\mathcal{BL}_i^a$ (Beliefs about beliefs).

- **Add belief data structures**: Belief table BBT^a:

Agent	Formula
Heli2	$\mathsf{in}(\mathtt{pos2}, \mathrm{Heli2} : GP(\mathtt{now}))$
Tank1	$\mathcal{B}_{\mathtt{Tank1}}(\mathrm{Heli1}, \mathsf{in}(\mathtt{pos1}, \mathrm{Heli1} : GP(\mathtt{now})))$
Tank2	$\mathcal{B}_{\mathtt{Tank2}}(\mathrm{Tank1}, \mathcal{B}_{\mathtt{Tank1}}(\mathrm{Heli1}, \mathsf{in}(\mathtt{pos4}, \mathrm{Heli1} : GP(\mathtt{now}))))$

  Belief semantics table BSemT^a: pairs
  $\langle h_1, \mathsf{Sem}_{feas}\rangle, \langle h_2, \mathsf{Sem}_{rat}\rangle, \langle h_3, \mathsf{Sem}_{reas}\rangle$.

MAP's and belief status sets

- **Meta AP**: AP + Belief Literals
- **Belief SS**: SS + Belief Atoms
- **Feasible Belief SS**: additional properties (Local feasibility, Compatibility with Belief Tables).
- **Implementing MAP's**: Beliefs as extended code calls

  $\mathcal{B}_a(c, \chi)$ corresponds to $\mathbf{in}(\langle c, \chi, \mathbf{true}\rangle, a : \mathit{belief\_table}())$

Compatible with
- Belief Sem.
- Belief Table

$$\begin{tikzcd} \mathcal{BP} \arrow[r, "\mathfrak{Trans}"] & \mathcal{P} \\ \mathcal{BS} \arrow[u, "\mathrm{Sem}^{new}"'] \arrow[r, "\mathfrak{Trans}"] & \mathcal{S} \arrow[u, "\mathcal{IC}^{ext}\ \text{Closure}", "\mathrm{Sem}^{old}"'] \end{tikzcd}$$

Temporal Annotations

- **Temporal Extents**: Timed Action. Checkpoints. Timed Effect Triple $\langle \{chk\}, Add, Del \rangle$. *Update altitude of the plane every 10 seconds.*

- **Scheduling**: Actions to be taken in the future. Temporal Annotations (ta) $[\mathsf{tai}_1, \mathsf{tai}_2]$: $[\mathtt{X}+\mathtt{Y}, \mathtt{X}+4\mathtt{y}]$. Add ta's to bodies of rules of agent programs:

$$\mathbf{in}(\mathtt{X}, \mathtt{satellite1} : broadcastData(\mathtt{GPS})) : [\mathtt{X}_{\mathsf{now}} - 5, \mathtt{X}_{\mathsf{now}}]$$

Temporal Agent Programs

- **Reasoning about the Past**: Temporal agent rules

$$\mathsf{Op}\,\alpha : [\mathsf{tai}_1, \mathsf{tai}_2] \quad \leftarrow \quad \varrho_1 : \mathsf{ta}_1 \wedge \cdots \wedge \varrho_n : \mathsf{ta}_n$$

  ($\mathsf{Op} \in \{\mathbf{P}, \mathbf{Do}, \mathbf{F}, \mathbf{O}, \mathbf{W}\}$, $\varrho_1 : \mathsf{ta}_1, \ldots, \varrho_n : \mathsf{ta}_n$ are tascs).

  If the body-terms are true at time $\leq t_{now}$, then the head must be true at a time $\geq t_{now}$.

- **Temporal SS**: $\mathcal{TS}_{t_{now}} : \mathbb{N} \longrightarrow \{S \mid S \text{ a status set }\}$.

- **Feasible Temporal SS**: additional properties (Checkpoint consistency, history compatibility, ...).

- **Computation**: Least fixed-point operator working on temporal interval constraint annotated status atoms.

Different Kinds of Uncertainty

- **State**: Probabilistic Approach. A *probabilistic code call* associated with $\mathcal{S} : f(\mathrm{d}_1, \ldots, \mathrm{d}_n)$ when executed on state $\mathcal{O}$ returns as output, a set of random variables of type τ $(\Sigma_{o \in \mathbf{R}V} \wp(o) \leq 1)$.

- **When will Effects of Actions occur**: Uncertain durations. Add a probability distribution to checkpoints. Temporal feasible SS will be sets of possible agent states.

- **Belief about Agents**: Extend belief tables to probabilistic belief tables by adding another row describing a probability distribution over belief formulae.

Probabilistic Agent Programs

- **Prob. Code Call Atoms**: $o \models_{\mathcal{O}}^{[\ell,u]} \mathbf{in}(\mathtt{X}, \mathcal{S} : f(\mathtt{d}_1, \ldots, \mathtt{d}_n))$ if and only if $(Y, \wp)$ is an output random variable when $\mathcal{S} : f(\mathtt{d}_1, \ldots, \mathtt{d}_n)$ is evaluated w.r.t. $\mathcal{O}$ and $o \in Y$ and $\ell \leq \wp(o) \leq u$.

- **Temporal agent rules**:

$$\mathsf{Op}_1\, \alpha : \quad \leftarrow \quad \Gamma, \mathsf{Op}_2\, \beta, \ldots, \mathsf{Op}_3\, \delta$$

(Op$_i$ $\in \{\mathbf{P}, \mathbf{Do}, \mathbf{F}, \mathbf{O}, \mathbf{W}\}$, Γ conjunction of annotated code call conditions $\chi : \langle[\mathsf{ai}_1, \mathsf{ai}_2], \otimes\rangle$).

$\chi : \langle[\mathsf{ai}_1, \mathsf{ai}_2], \otimes\rangle$ means: probability of χ being true lies in the interval defined by $[\mathsf{ai}_1, \mathsf{ai}_2]$.

- **Probabilistic SS**: Same as SS.
- **Feasible Probabilistic SS**: Same conditions as for AP's, but extended to annotated code calls. Actions fire if preconditions are true with probability 1.
- **Computation**: Least fixed-point operator working on status sets and probabilistic agent programs.

References

Funding: By DARPA and ARL (4 Mio $, Hendler/V.S.).

URL: `http://www.cs.umd.edu/users/vs/agent/impact.html`

Papers: **AI**: **Agent Programs** I.//II. (Eiter/Pick/VS).

IEEE, JACM: **Overall Descriptions** (VS/Hendler).

JLP: **Metaknowledge** (Dix/Pick/VS).

Submitted (available as TR's):

1. **Temporal Programs** (Dix/Kraus/Subrahmanian),
2. **Uncertainty** (Dix/Kraus/Subrahmanian),
3. **Security** (Bonatti/Kraus/Subrahmanian).

Book: appears at the end of 1999/2000 at MIT-Press.