



Designing Goal-Oriented Reactive Behaviours

(experiences and some design guidelines)

Peter Novák and Michael Köster

Clausthal University of Technology, Germany

July 21th, 2008

CogRob 2008, Patras, Greece

Outline

- 1 Motivation
- 2 Behavioural State Machines
- 3 Jazzbot
- 4 Design and implementation
 - Knowledge representation
 - Behaviours
 - Control cycle
- 5 Conclusion

Problem

Different programming languages are suitable for different knowledge representation tasks.



Heterogeneous knowledge bases!

Relationships between agent's KBs lost!

Desiderata:

- 1 heterogeneous KR \rightsquigarrow loose agent architecture
- 2 hybrid architecture \rightsquigarrow reactivity + deliberation
- 3 flexible and practical computation model \rightsquigarrow syntax + semantics

Problem

Different programming languages are suitable for different knowledge representation tasks.



Heterogeneous knowledge bases!

Relationships between agent's KBs lost!

Desiderata:

- 1 heterogeneous KR \rightsquigarrow loose agent architecture
- 2 hybrid architecture \rightsquigarrow reactivity + deliberation
- 3 flexible and practical computation model \rightsquigarrow syntax + semantics

Problem

Different programming languages are suitable for different knowledge representation tasks.



Heterogeneous knowledge bases!

Relationships between agent's KBs lost!

Desiderata:

- 1 heterogeneous KR \rightsquigarrow loose agent architecture
- 2 hybrid architecture \rightsquigarrow reactivity + deliberation
- 3 flexible and practical computation model \rightsquigarrow syntax + semantics

Focus \rightsquigarrow agent's behaviours

Behavioural State Machines

A *lightweight* programming framework with clear separation between *knowledge representation* and agent's *behaviours*.

BSM agent: $\mathcal{A} = (\mathcal{M}_1, \dots, \mathcal{M}_n, \mathcal{P})$

KR module $\mathcal{M} = (\mathcal{S}, \mathcal{L}, \mathcal{Q}, \mathcal{U})$

- \mathcal{S} - a set of states
- \mathcal{L} - a KR language,
- \mathcal{Q} - a set of query operators $\models: \mathcal{S} \times \mathcal{L} \rightarrow \{\top, \perp\}$,
- \mathcal{U} - set of update operators $\oplus: \mathcal{S} \times \mathcal{L} \rightarrow \mathcal{S}$.

mental state transformer $\tau: \models_i \varphi \longrightarrow \oplus_j \psi$

transition system over states $\sigma = \langle \sigma_1, \dots, \sigma_n \rangle$ induced by updates $\oplus \psi$
yielded by the agent program \mathcal{P}

Pragmatics:

- high level syntax: nested mst's
- re-usable code: macro preprocessor

BSM agent: $\mathcal{A} = (\mathcal{M}_1, \dots, \mathcal{M}_n, \mathcal{P})$

KR module $\mathcal{M} = (\mathcal{S}, \mathcal{L}, \mathcal{Q}, \mathcal{U})$

- \mathcal{S} - a set of states
- \mathcal{L} - a KR language,
- \mathcal{Q} - a set of query operators $\models: \mathcal{S} \times \mathcal{L} \rightarrow \{\top, \perp\}$,
- \mathcal{U} - set of update operators $\oplus: \mathcal{S} \times \mathcal{L} \rightarrow \mathcal{S}$.

mental state transformer $\tau: \models_i \varphi \longrightarrow \oplus_j \psi$

transition system over states $\sigma = \langle \sigma_1, \dots, \sigma_n \rangle$ induced by updates $\oplus \psi$
yielded by the agent program \mathcal{P}

Pragmatics:

- high level syntax: nested mst's
- re-usable code: macro preprocessor

BSM agent: $\mathcal{A} = (\mathcal{M}_1, \dots, \mathcal{M}_n, \mathcal{P})$

KR module $\mathcal{M} = (\mathcal{S}, \mathcal{L}, \mathcal{Q}, \mathcal{U})$

- \mathcal{S} - a set of states
- \mathcal{L} - a KR language,
- \mathcal{Q} - a set of query operators $\models: \mathcal{S} \times \mathcal{L} \rightarrow \{\top, \perp\}$,
- \mathcal{U} - set of update operators $\oplus: \mathcal{S} \times \mathcal{L} \rightarrow \mathcal{S}$.

mental state transformer $\tau: \models_i \varphi \longrightarrow \oplus_j \psi$

transition system over states $\sigma = \langle \sigma_1, \dots, \sigma_n \rangle$ induced by updates $\oplus \psi$ yielded by the agent program \mathcal{P}

Pragmatics:

- **high level syntax:** nested mst's
- **re-usable code:** macro preprocessor

How to program agents with BSM?

	BSM	vs.	AgentSpeak(L) et al.
architecture	modular, multiple KRs application specific		fixed, single KR
reactivity	reactivity, then deliberation adjustable deliberation		deliberation, then reactivity reactive planning

Goal-centric design:

- goals vs. beliefs \rightsquigarrow goal as a *context holder*
- goals vs. behaviours \rightsquigarrow goal as a *behaviour trigger*

\rightsquigarrow Goal-Oriented Reactive Behaviours

How to program agents with BSM?

	BSM	vs.	AgentSpeak(L) et al.
architecture	modular, multiple KRs application specific		fixed, single KR
reactivity	reactivity, then deliberation adjustable deliberation		deliberation, then reactivity reactive planning

Goal-centric design:

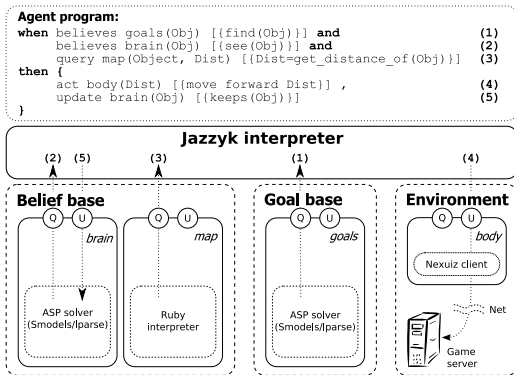
- goals vs. beliefs \rightsquigarrow goal as a *context holder*
- goals vs. behaviours \rightsquigarrow goal as a *behaviour trigger*

\rightsquigarrow **Goal-Oriented Reactive Behaviours**

Jazzbot

Softbot in a simulated 3D environment - Nexuiz game

- challenging, dynamic and rich environment



Environment & beliefs - model of the world

Environment module: $\mathcal{E} = (\mathcal{L}_{Nexviz}, \{\models_{\mathcal{E}}\}, \{\odot_{\mathcal{E}}\})$

query $\models_{\mathcal{E}}$: sensors

update $\odot_{\mathcal{E}}$: actuators

Belief base: $\mathcal{B} = (AnsProlog^*, \{\models_{\mathcal{B}}\}, \{\oplus_{\mathcal{B}}, \ominus_{\mathcal{B}}\})$

- logic program \rightsquigarrow bot's beliefs about the environment, its body, moods, ...
- reflects the state of \mathcal{E}

Environment & beliefs - model of the world

Environment module: $\mathcal{E} = (\mathcal{L}_{Nexviz}, \{ \models_{\mathcal{E}} \}, \{ \odot_{\mathcal{E}} \})$

query $\models_{\mathcal{E}}$: sensors

update $\odot_{\mathcal{E}}$: actuators

Belief base: $\mathcal{B} = (AnsProlog^*, \{ \models_{\mathcal{B}} \}, \{ \oplus_{\mathcal{B}}, \ominus_{\mathcal{B}} \})$

- **logic program** \rightsquigarrow bot's beliefs about the environment, its body, moods, ...
- reflects the state of \mathcal{E}

Goal base

- types:** *declarative goals* (to-be) vs. *tasks* (to-do)
- commitments:** maintain/achieve/avoid, blind/single-/open- minded
- interactions:** goal/subgoals, task/subtasks, integrity constraints

Goal base: $\mathcal{G} = (AnsProlog^*, \{\models_g\}, \{\oplus_g, \ominus_g\})$

- **logic program** \rightsquigarrow stores bot's desires
- interaction between *declarative goals* vs. *tasks*

$$task(\dots) \leftarrow achieve(\dots)$$

- integrity constraints

$$\perp \leftarrow task(\dots) \wedge task(\dots)$$

Goal base

- types:** *declarative goals* (to-be) vs. *tasks* (to-do)
- commitments:** maintain/achieve/avoid, blind/single-/open- minded
- interactions:** goal/subgoals, task/subtasks, integrity constraints

Goal base: $\mathcal{G} = (AnsProlog^*, \{ \models_{\mathcal{G}} \}, \{ \oplus_{\mathcal{G}}, \ominus_{\mathcal{G}} \})$

- **logic program** \rightsquigarrow stores bot's desires
- interaction between *declarative goals* vs. *tasks*

$$task(\dots) \leftarrow achieve(\dots)$$

- integrity constraints

$$\perp \leftarrow task(\dots) \wedge task(\dots)$$

Behavioural layer

encoding interactions between KR modules

Information flow between the KR modules:

- 1 environment \longrightarrow beliefs *(perception)*
- 2 beliefs \longrightarrow goals *(commitment strategies)*
- 3 goals \longrightarrow environment *(behaviour selection)*

Perceptions

PERCEIVE: $\models_{\varepsilon} \varphi \longrightarrow \bigcirc_{\mathcal{B}} \psi$

- mapping of information from the environment into the belief base
 - assert** \rightsquigarrow insert a fact, remember
 - retract** \rightsquigarrow delete a fact, forgetting

Commitment strategies

HANDLE_GOALS: $\models_B \varphi \longrightarrow \mathcal{O}_G \psi$

- goal commitment strategies **encoded explicitly** - different goals have different strategies
 - goal adoption
 - dropping a goal
- updates modify **only to-be part** the goal base:
achieve/maintain/avoid(...)

Action selection

ACT: $\models_{\mathcal{G}} \varphi \wedge \phi_{\mathcal{B}} \longrightarrow \tau$

- goals trigger behaviours
 - exogeneous behaviours: $\phi_{\mathcal{G}} \wedge \phi_{\mathcal{B}} \longrightarrow \mathcal{O}_{\mathcal{E}}\psi$
- queries check only *to-do* part of the goal base: $task(\dots)$



Control models

- choice of operators for joining individual mst's:

PERCEIVE ◦ HANDLE_GOALS ◦ ACT

vs.

PERCEIVE | HANDLE_GOALS | ACT

- similarly for nested mst's:
 - check a single sensor vs. all the sensors
 - drop a single satisfied goal vs. all the satisfied goals
 - execute a single behaviour vs. a sequence of behaviours

Conclusion: methodology guidelines

We assume a $\mathcal{E}BD\mathcal{E}$ architecture (fixed \mathcal{E}):

- 1 \mathcal{G} : declarative goals, corresponding tasks and their interactions
- 2 τ_{act} : behaviours corresponding to the tasks
- 3 τ_{cs} : adopt/drop conditions for goals \rightsquigarrow commitment strategies
- 4 \mathcal{B} : beliefs corresponding to the goals and their interactions
- 5 τ_{perc} : link beliefs to sensory input
- 6 BSM \mathcal{A} : execution strategy for mst's τ_{perc} , τ_{cs} , τ_{act} , global control cycle

Conclusion: methodology guidelines

We assume a $\mathcal{E}BD\mathcal{E}$ architecture (fixed \mathcal{E}):

- 1 \mathcal{G} : declarative **goals**, corresponding **tasks** and their interactions
- 2 τ_{act} : behaviours corresponding to the tasks
- 3 τ_{cs} : adopt/drop conditions for goals \rightsquigarrow commitment strategies
- 4 \mathcal{B} : beliefs corresponding to the goals and their interactions
- 5 τ_{perc} : link beliefs to sensory input
- 6 BSM \mathcal{A} : execution strategy for mst's τ_{perc} , τ_{cs} , τ_{act} , global control cycle

Conclusion: methodology guidelines

We assume a $\mathcal{E}BD\mathcal{E}$ architecture (fixed \mathcal{E}):

- 1 \mathcal{G} : declarative **goals**, corresponding **tasks** and their interactions
- 2 τ_{act} : **behaviours** corresponding to the tasks
- 3 τ_{cs} : adopt/drop conditions for goals \rightsquigarrow commitment strategies
- 4 \mathcal{B} : beliefs corresponding to the goals and their interactions
- 5 τ_{perc} : link beliefs to sensory input
- 6 BSM \mathcal{A} : execution strategy for mst's τ_{perc} , τ_{cs} , τ_{act} , global control cycle

Conclusion: methodology guidelines

We assume a $\mathcal{E}BD\mathcal{E}$ architecture (fixed \mathcal{E}):

- 1 \mathcal{G} : declarative **goals**, corresponding **tasks** and their interactions
- 2 τ_{act} : **behaviours** corresponding to the tasks
- 3 τ_{cs} : adopt/drop conditions for goals \rightsquigarrow commitment strategies
- 4 \mathcal{B} : beliefs corresponding to the goals and their interactions
- 5 τ_{perc} : link beliefs to sensory input
- 6 BSM \mathcal{A} : execution strategy for mst's τ_{perc} , τ_{cs} , τ_{act} , global control cycle

Conclusion: methodology guidelines

We assume a $\mathcal{E}BD\mathcal{E}$ architecture (fixed \mathcal{E}):

- 1 \mathcal{G} : declarative **goals**, corresponding **tasks** and their interactions
- 2 τ_{act} : **behaviours** corresponding to the tasks
- 3 τ_{cs} : adopt/drop conditions for goals \rightsquigarrow commitment strategies
- 4 \mathcal{B} : beliefs corresponding to the goals and their interactions
- 5 τ_{perc} : link beliefs to sensory input
- 6 BSM \mathcal{A} : execution strategy for mst's τ_{perc} , τ_{cs} , τ_{act} , global control cycle

Conclusion: methodology guidelines

We assume a $\mathcal{E}BD\mathcal{E}$ architecture (fixed \mathcal{E}):

- 1 \mathcal{G} : declarative **goals**, corresponding **tasks** and their interactions
- 2 τ_{act} : **behaviours** corresponding to the tasks
- 3 τ_{cs} : adopt/drop conditions for goals \rightsquigarrow commitment strategies
- 4 \mathcal{B} : beliefs corresponding to the goals and their interactions
- 5 τ_{perc} : link beliefs to sensory input
- 6 BSM \mathcal{A} : execution strategy for mst's τ_{perc} , τ_{cs} , τ_{act} , global control cycle

Conclusion: methodology guidelines

We assume a $\mathcal{E}BD\mathcal{E}$ architecture (fixed \mathcal{E}):

- 1 \mathcal{G} : declarative **goals**, corresponding **tasks** and their interactions
- 2 τ_{act} : **behaviours** corresponding to the tasks
- 3 τ_{cs} : adopt/drop conditions for goals \rightsquigarrow commitment strategies
- 4 \mathcal{B} : beliefs corresponding to the goals and their interactions
- 5 τ_{perc} : link beliefs to sensory input
- 6 **BSM** \mathcal{A} : execution strategy for mst's τ_{perc} , τ_{cs} , τ_{act} , **global control cycle**



Ongoing & future work

1 towards formal specifications

- re-usable code templates with a clear semantics
 - ↪ composition of macros
- support of analysis stage
- verification (partial?)

↪ *dynamic linear time temporal logics*

2 further case-studies

- videogames (Jazzbot)
- entertainment robotics (Urbi)
- test-bed for KRs (Prolog, LISP/Scheme)
- multi-agent coordination



Thank you for your attention.

<http://jazzyk.sourceforge.net/>