Programming with Goals (1)

M. Birna van Riemsdijk, TU Delft, The Netherlands

Part of slides adapted from MAS course slides by Hindriks

4/24/11

Overview

- Representation of goals in agent programming languages
- GOAL mental states
- Goal types

1.

Representing Goals in Agent Programming Languages

Agent Programming Languages

- 1993: AGENT-0 (Shoham)
- 1996: **AgentSpeak(L)** (Rao; inspired by PRS)
- 1996: Golog (Reiter, Levesque, Lesperance)
- 1997: 3APL (Hindriks et al.)
- 1998: ConGolog (Giacomo, Levesque, Lesperance)
- 2000: **JACK** (Busetta, Howden, Ronnquist, Hodgson)
- 2000: **GOAL** (Hindriks et al.)
- 2000: CLAIM (Amal El FallahSeghrouchni)
- 2002: **Jason** (Bordini, Hubner; implementation of AgentSpeak)
- 2003: **Dribble** (Van Riemsdijk et al.; combination of 3APL & GOAL)
- 2003: **Jadex** (Braubach, Pokahr, Lamersdorf)
- 2008: **2APL** (Dastani et al., successor of 3APL)

This overview is far from complete!

Achievement Goals

- Implemented cognitive agent programming languages typically incorporate achievement goals
- Achievement goal: goal to reach a certain state of affairs e.g., be at a certain location, have a weapon, have a clean floor, have a block on top of another block
- Declarative goal
- Different ways of representing goals
- Different semantics for goals

Jason – achievement goals (1)

http://jason.sourceforge.net/Jason/Jason.html

```
+green_patch(Rock)
   :  not battery_charge(low)
   <- ?location(Rock,Coordinates);
      !at(Coordinates);
      !examine(Rock).

+!at(Coords)
   :  not at(Coords)
      & safe_path(Coords)
   <- move_towards(Coords);
      !at(Coords).

+!at(Coords) ...
```

achievement goal (creation)

achievement goal (plan trigger)

Jason - achievement goals (2)

- Represented as predicate !p(t1,...,tn)
- Used as plan triggers
- Created from within plans
- Stored as events in event base

Jadex – achievement goals

http://jadex.informatik.uni-hamburg.de/xwiki/bin/view/About/Overview

```
<goals>
    <achievegoal name="translate">
        <parameter name="direction" class="String"/>
        <parameter name="word" class="String"/>
        <parameter name="result" class="String" direction="out"/>
    </achievegoal>
</goals>
```

- Specified in XML
- Used as plan triggers
- Created from within plans in Java

```
IGoal goal = createGoal("translate");...;
dispatchSubgoalAndWait(goal);
```

- Stored as objects in goal base

GOAL - achievement goals

- Represented as conjunctions of atoms
  $p_1(t1,...,tn), ..., p_k(t1,...,tm)$
- Used for action selection
- Created from within action rules
- Stored in goal base

References

Websites

- 2APL: http://www.cs.uu.nl/2apl/
- Agent Factory: http://www.agentfactory.com
- GOAL: http://mmi.tudelft.nl/trac/goal
- JACK: http://www.agent-software.com.au/products/jack/
- Jadex: http://jadex.informatik.uni-hamburg.de/
- Jason: http://jason.sourceforge.net/
- JIAC: http://www.jiac.de/

Books

- Bordini, R.H.; Dastani, M.; Dix, J.; El Fallah Seghrouchni, A. (Eds.), 2005 Multi-Agent Programming Languages, Platforms and Applications. *presents 3APL, CLAIM, Jadex, Jason*
- Bordini, R.H.; Dastani, M.; Dix, J.; El Fallah Seghrouchni, A. (Eds.), 2009, Multi-Agent Programming: Languages, Tools and Applications. *presents a.o.: Brahms, CArtAgO, GOAL, JIAC Agent Platform*

2.

GOAL Mental States

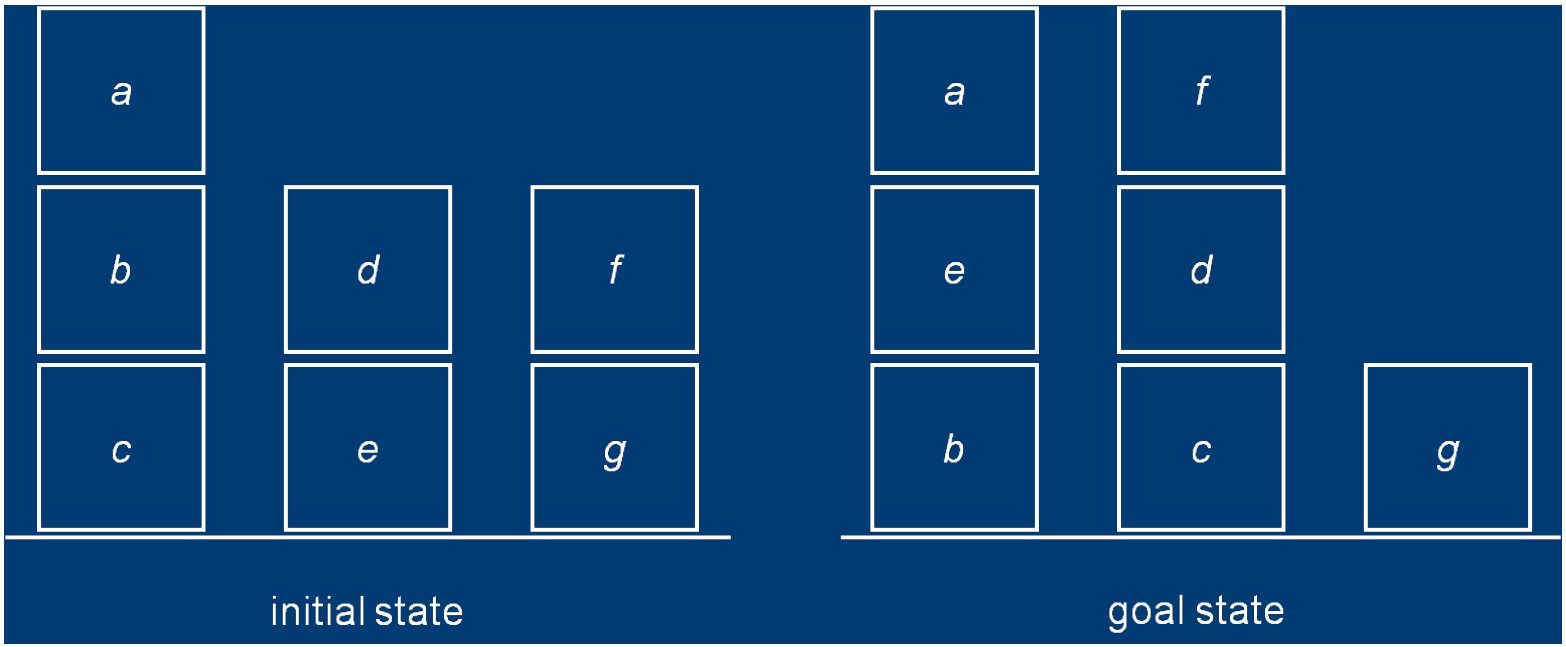
GOAL Mental State: Overview

- Beliefs
  represent current state of environment (Prolog)
- Knowledge
  represent (static) domain knowledge (Prolog)
- Goals
  represent achievement goals (conjunctions of atoms)

The Blocks World

A classic AI planning *problem.*

Objective: Move blocks in initial state such that result is goal state.



- *Positioning* of blocks on table is not relevant.
- A block can be moved *only if* it there is no other block on top of it.

Representing the Blocks World

***Prolog is the* knowledge representation language *used in* GOAL.**

Basic predicates:

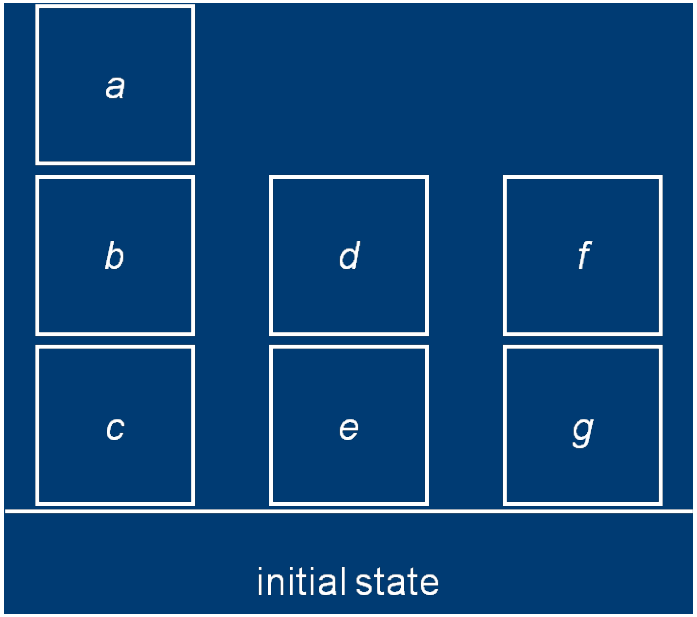
- `block(X).`
- `on(X,Y).`

Defined predicates:

- ```
  tower([X]) :- on(X,table).
  tower([X,Y|T]) :- on(X,Y),tower([Y|T]).
  ```
- `clear(X) :- block(X), not(on(Y,X)).`
  ```

Representing the Initial State

Using the* on(X,Y) *predicate we can represent the initial state.




```
beliefs{
   on(a,b),
   on(b,c),
   on(c,table),
   on(d,e),
   on(e,table),
   on(f,g),
   on(g,table).
}
```

Initial belief base of agent

Representing the Blocks World

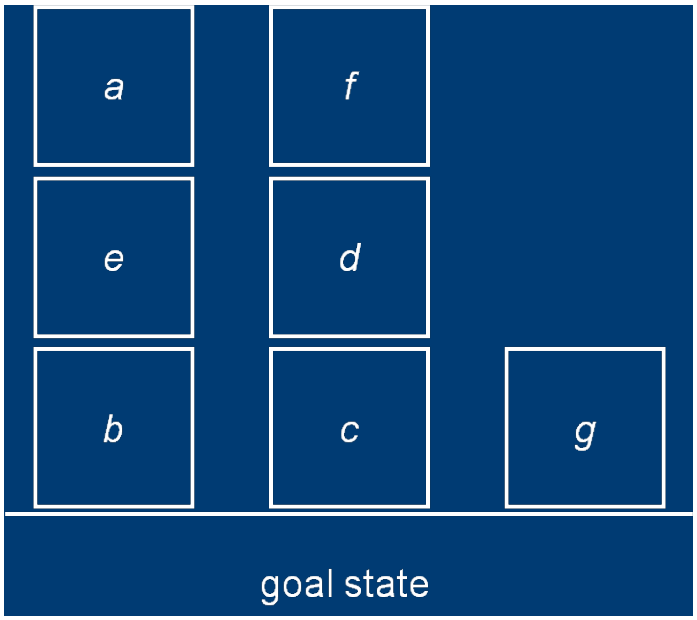
- What about the rules we defined before?
- Insert clauses that do not change into the knowledge base.

```
knowledge{
  block(X) :- on(X,Y).
  clear(X) :- block(X), not(on(Y,X)).
  clear(table).
  tower([X]) :- on(X,table).
  tower([X,Y|T]) :- on(X,Y), tower([Y|T]).
}
```

Static knowledge base of agent

Representing the Goal State

Using the* on(X,Y) *predicate we can represent the goal state.




```
goals{
  on(a,e),
  on(b,table),
  on(c,table),
  on(d,c),
  on(e,b),
  on(f,d),
  on(g,table).
}
```

Initial goal base of agent

One or Many Goals

In the goal base using the comma- or period-separator makes a difference!

```
goals{
   on(a,table).
   on(b,a).
   on(c,b).
}
```

```
goals{
   on(a,table),
   on(b,a),
   on(c,b).
}
```

- Left goal base has three goals, right goal base has single goal.
- Single goal: conjuncts have to be achieved at the same time

Mental State of GOAL Agent

```
knowledge{
  block(X) :- on(X,_).
  clear(X) :- block(X), not(on(Y,X)).
  clear(table).
  tower([X]) :- on(X,table).
  tower([X,Y|T]) :- on(X,Y), tower([Y|T]).
}
beliefs{
  on(a,b), on(b,c), on(c,table), on(d,e), on(e,table),
  on(f,g), on(g,table).
}
goals{
  on(a,e), on(b,table), on(c,table), on(d,c), on(e,b),
  on(f,d), on(g,table).
}
```

Initial mental state of agent

Inspecting the Belief & Goal Base

- Operator `bel(φ)` to inspect the belief base.
- Operator `goal(φ)` to inspect the goal base.
  - Where φ is a Prolog conjunction of literals.
- Examples:
  - `bel(clear(a), not(on(a,c))).`
  - `goal(tower([a,b])).`

Inspecting the Belief Base

- `bel(φ)` succeeds if φ follows from the *belief base in combination with the knowledge base*.

```
knowledge{
  block(X) :- on(X,_).
  clear(X) :- block(X), not(on(Y,X)).
  clear(table).
  tower([X]) :- on(X,table).
  tower([X,Y|T]) :- on(X,Y), tower([Y|T]).
}
beliefs{
  on(a,b), on(b,c), on(c,table), on(d,e), on(e,table),
  on(f,g), on(g,table).
}
```

- Example: `bel(clear(a), not(on(a,c)))` succeeds
- Condition φ is evaluated as a Prolog query.

Inspecting the Goal Base

Use the* goal(...) *operator to inspect the goal base.

- `goal(φ)` succeeds if φ follows from one of the goals in the goal base *in combination with the knowledge base*.

```
knowledge{
   block(X) :- on(X,_).
   clear(X) :- block(X), not(on(Y,X)).
   clear(table).
   tower([X]) :- on(X,table).
   tower([X,Y|T]) :- on(X,Y), tower([Y|T]).
}
goals{
  on(a,e), on(b,table), on(c,table), on(d,c), on(e,b),
   on(f,d), on(g,table).
}
```

- Example: `goal(clear(a))` succeeds. but not `goal(clear(a),clear(c)).`

Why a Separate Knowledge Base?

- Concepts defined in KB can be used in combination with both the belief and goal base.
- *Example*
  - Since agent believes `on(e,table),on(d,e)` infer: agent believes `tower([d,e]).`
  - If agent wants `on(a,table),on(b,a)` infer: agent wants `tower([b,a]).`
- Knowledge base introduced to avoid duplicating clauses in belief and goal base.

Combining Beliefs and Goals

Useful to combine the `bel(…)` and `goal(…)` operators.

- Achievement goals
  - **`a-goal(φ) = goal(φ), not(bel(φ))`**
    - Agent only has an achievement goal if it does not believe the goal has been reached already.
    - E.g., if belief base is {p.} and goal base is {p,q.}, a-goal(q) but not a-goal(p) holds
- Goal achieved
  - **`goal-a(φ) = goal(φ), bel(φ)`**
    - A (sub)-goal φ has been achieved if the agent believes φ.

Features of Goals in GOAL

- Goals stored in separate goal base
- Goals are conjunctions of atoms
- Goals can be inspected using the operators:
  `goal, a-goal and goal-a`
- Goals are derived from one of the goals in goal base in combination with knowledge base

3.

Goal Types

Goal Types

- Achievement goals most common
- Simple and intuitive meaning and implementation
- But... not the only goal type that one may want to use

Jadex - Goal Types (2004)

L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf. Goal representation for BDI agent systems. In ProMAS'04, volume 3346 of LNAI, pages 44–65. Springer, Berlin, 2005

- Achieve goal: specifies world state an agent wants to bring about; *e.g., cleanup*
- Maintain goal: observe desired world state and agent should actively try to re-establish this state when it is violated; *e.g., batteryLoaded*
- Perform goal: specifies activities to be done; outcome of the goal depends only on whether activities were performed; *e.g., lookForWaste*
- Query goal: enquire information about a specified issue; *e.g., queryWasteBin*

Goal Types: Formalization (2006)

M. Dastani, M. B. van Riemsdijk, and J.-J. Ch. Meyer. Goal types in agent programming. In Proc. of ECAI'06, volume 141 of Frontiers in Artificial Intelligence and Applications, pages 220–224. IOS Press, 2006.

- Achieve goal: achieve certain state of affairs
- Maintain goal: ensure that a state holds and continues to hold; plans should be generated and executed if the state denoted by the maintain goal is threatened not to hold
- Perform goal: generate plans without requiring that they have desired result (compare achievement goal Jason)

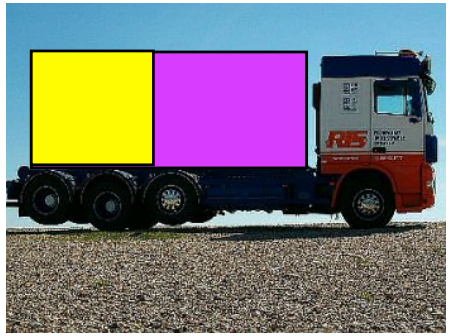
Satisfying Maintenance Goals (2007)

K. Hindriks and M. B. van Riemsdijk. Satisfying maintenance goals. In Proc. of DALT'07, 2007.

- Aim: prevent maintenance goal violation by constraining actions that agent can execute
- Means: lookahead mechanism to prevent choosing paths that will lead to violation
- Extension of GOAL with maintenance goals (not implemented)

Example: Carrier Agent

A

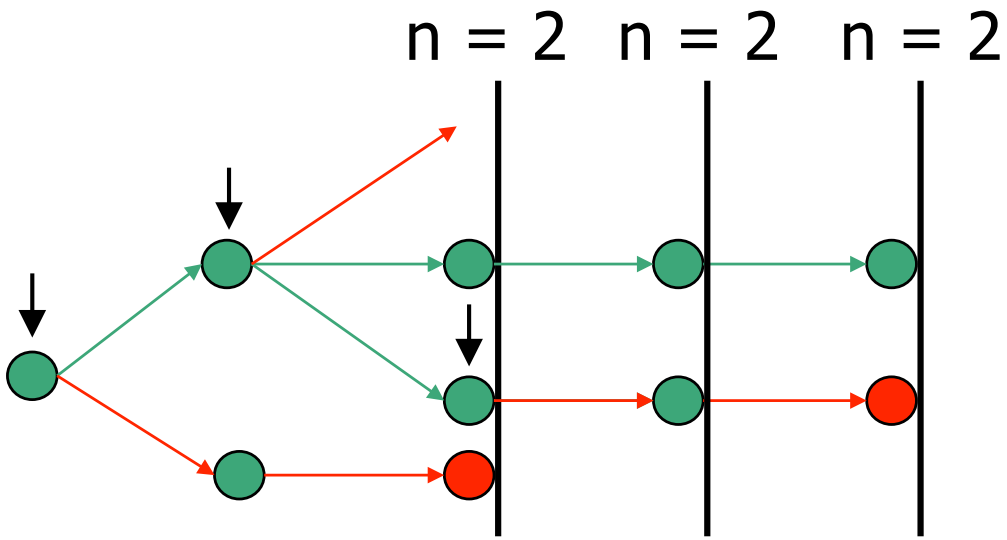


B

- bring parcels from A to B
- need refueling: maintenance goal "gas > 0"
  constrains action of driving, refuel to actively prevent violation
- lookahead: how far is the next gas station?

- not overloading truck: maintenance goal "weight < 20"
  conflict between maintenance goals and achievement goals

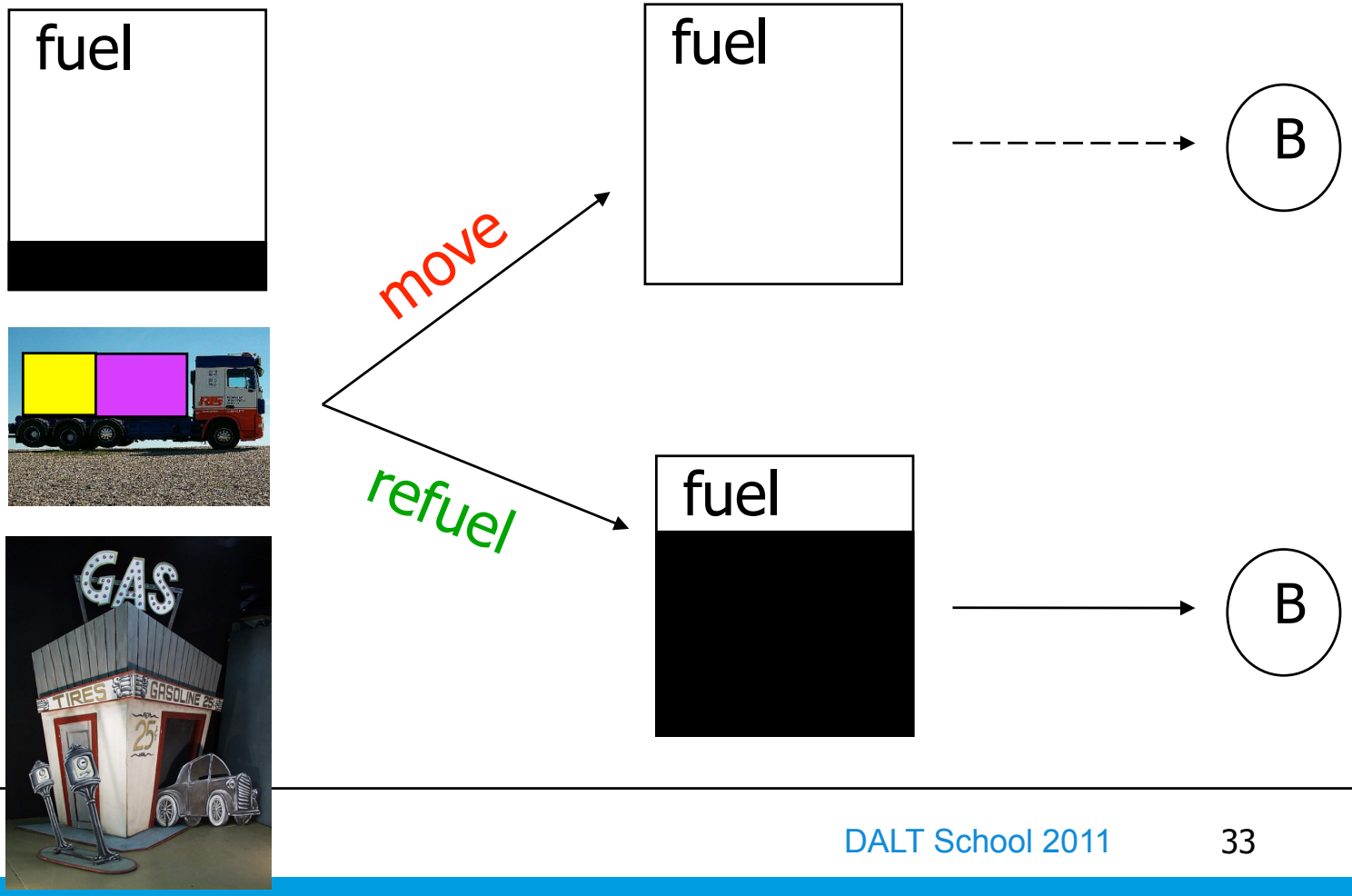
Lookahead



n-step lookahead: agent should not enter a path that leads to a violation of maintenance goals within n steps

Example: Carrier Agent

Maintenance goal: fuel > 0



Linear Temporal Logic (LTL)

Evaluated over traces

- $\Diamond\chi$ eventually: X should hold sometime in the future
- $\bigcirc\chi$ next: X should hold in the next state
- $\chi^{U}\chi_2$ until: X should hold until X_2
- $\Box\chi \equiv \neg\Diamond\neg\chi$ always: X should hold on every state

achievement goal ϕ : $\Diamond\phi$

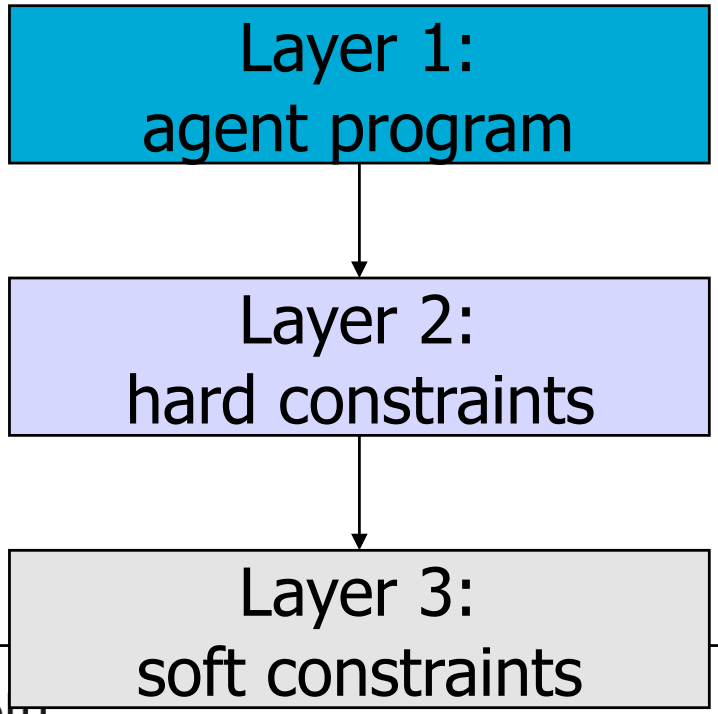
maintenance goal ϕ : $\Box\phi$

Temporal Logic for Integrating Goals and Preferences (2008)

K. Hindriks and M. B. van Riemsdijk. Using temporal logic to integrate goals and qualitative preferences into agent programming. In Proc. of DALT'08, 2009.

Rational Action Selection Architecture (RASA)

- three-tier architecture + temporal logic
- separation of concerns, added flexibility



generate options for action using action selection rules

reduce options through hard constraints

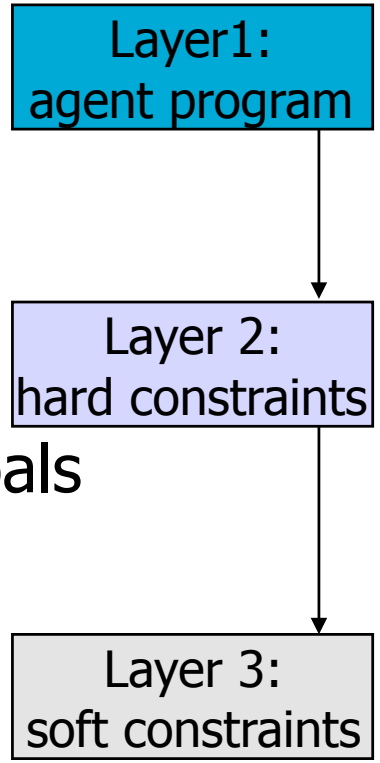
choose a most preferred option

RASA

- Layer 1: GOAL
  beliefs - propositional formulas
  goals - LTL formulas

- Layer 2: goals as hard constraints
  lookahead: select paths without violation of goals

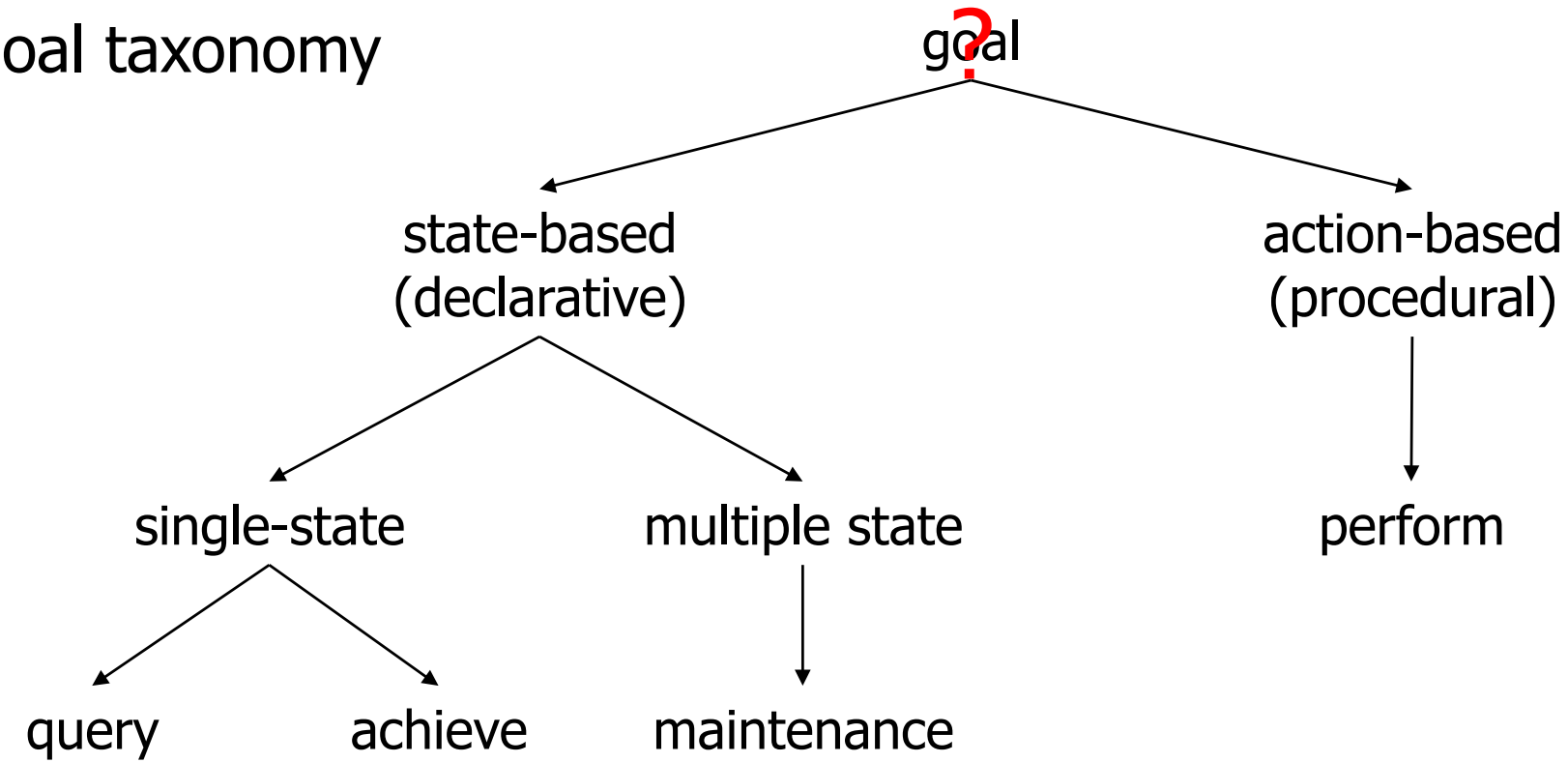
- Layer 3: soft constraints
  preferences: sequence of LTL formulas
  induce lexicographic preference ordering over paths



Goals: A Unifying Framework

M. B. van Riemsdijk, M. Dastani, and M. Winikoff. Goals in agent systems: A unifying framework. In Padgham, Parkes, Müller, and Parsons, editors, *Autonomous Agents and Multi-Agent Systems (AAMAS), pages 713–720. IFAAMAS, 2008.*

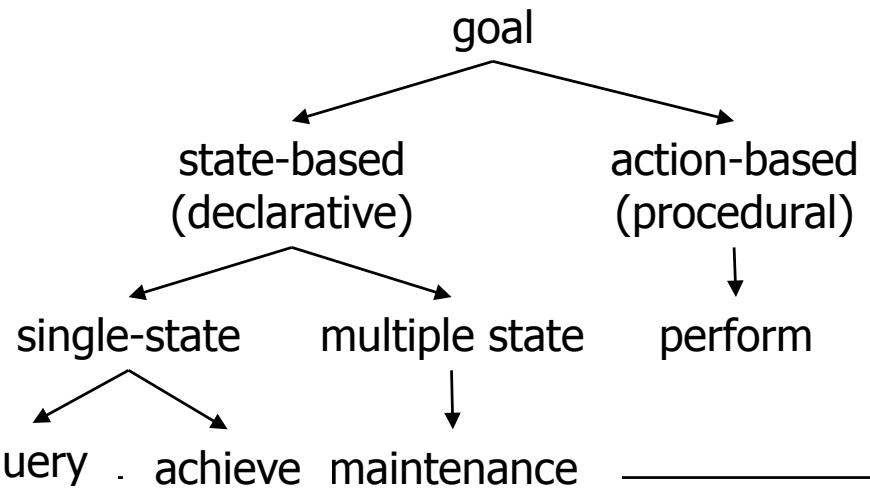
Goal taxonomy



Definition of Goal

A goal is *a mental attitude representing preferred progressions of a particular multi-agent system.... that the agent has chosen to put effort into bringing about.*

progression = a passing successively from one member of a series to the next



Summary

- Representation of goals
  - XML, atoms, conjunctions of atoms
- GOAL mental states
  - beliefs, goals, knowledge
- Goal types
  - achieve, maintain, perform, query
  - goals as LTL formulas