Agent Reasoning: Knowledge, Plans and Flexible Control

Francesca Toni

Department of Computing, Imperial College London, UK

DALT School 2011
Bertinoro, Italy

10-15 April, 2011

Outline

I. Agents
   - Logic- and LP-based approaches
   - KGP agents

II. Multi-agent systems
   - Communication
   - (Negotiation)

III. Argumentative (KGP) agents
   - Service-oriented architectures and Grid

Part I

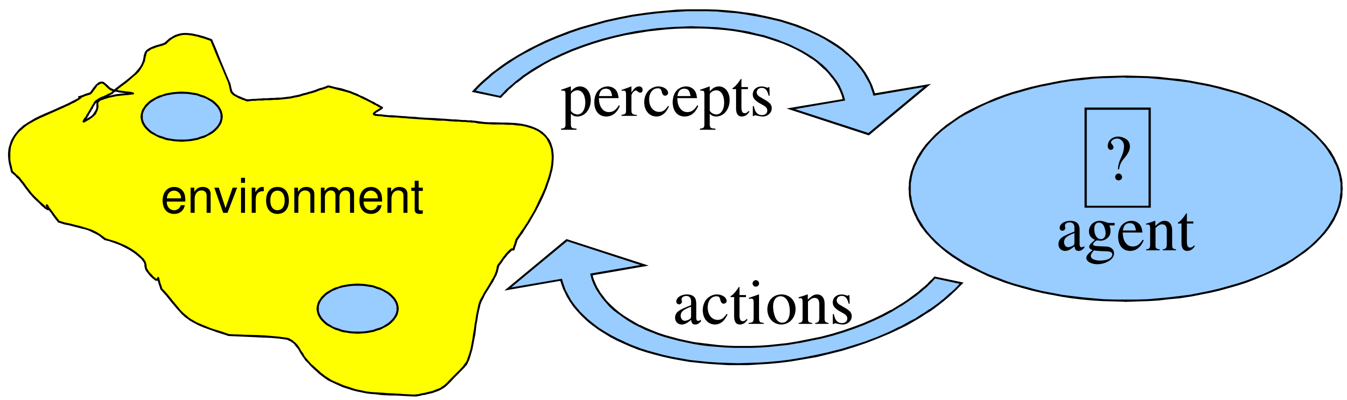
Agents

- General Introduction
- Logic- and LP-based approaches
- KGP agents

Agents

Autonomous systems that

- Perceive the environment in which they are situated (via sensors)
- Act upon the environment (via effectors)



- designed with certain “performance” requirements
  - maintain environment in a certain state
  - achieve certain state of its environment

Russel& Norvig, AI: a modern approach

Agents (cntd)

In general terms agents are often defined as (software or hardware) entities that are at least

- *autonomous* (no centralised control)
- *reactive* (to the environment)
- *deliberative/pro-active* (towards goals)
- *social/interactive* (via observation + communication with other agents in a multi-agent system)

Wooldridge, *An introduction to multiagent systems*

Autonomy

- To avoid problems with centralised control:
  - Complexity of the solution
  - Maintenance (especially in open systems)
  - Difficulty in tracking problems
- For applications whose components are by nature autonomous
  - electronic auctions, e-procurement .....

E.g. have a joint set of notes vs each lecturer prepares his/her own notes

Pro-activeness

- Agents have goals/objectives (e.g. be in Bertinoro on 10 April)
- Agents generate plans for the achievement of their goals
  - "Partial" plans (e.g. book flight and then arrange stay (e.g. taxi+hotel or car+agriturismo)
  - Partially instantiated plans (e.g. book hotel X in Bertinoro and transport to hotel X)
  - Re-planning (e.g. booked flight is cancelled)
  - Cooperative planning (e.g. share a taxi to stay within budget)

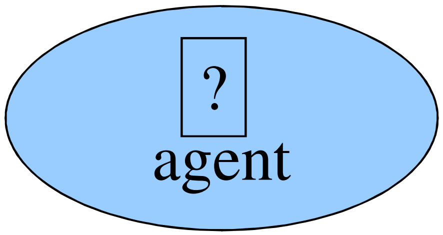
Reactivity

- Traditional AI: systems assuming
  - complete and correct knowledge of the world
  - goals and knowledge fixed at design time
- Reactivity to cope with
  - Incomplete/possibly incorrect information at design time (e.g. gate at airport/weather forecast for Bertinoro)
  - dynamically changing goals (e.g. get dinner/go to gate) and knowledge

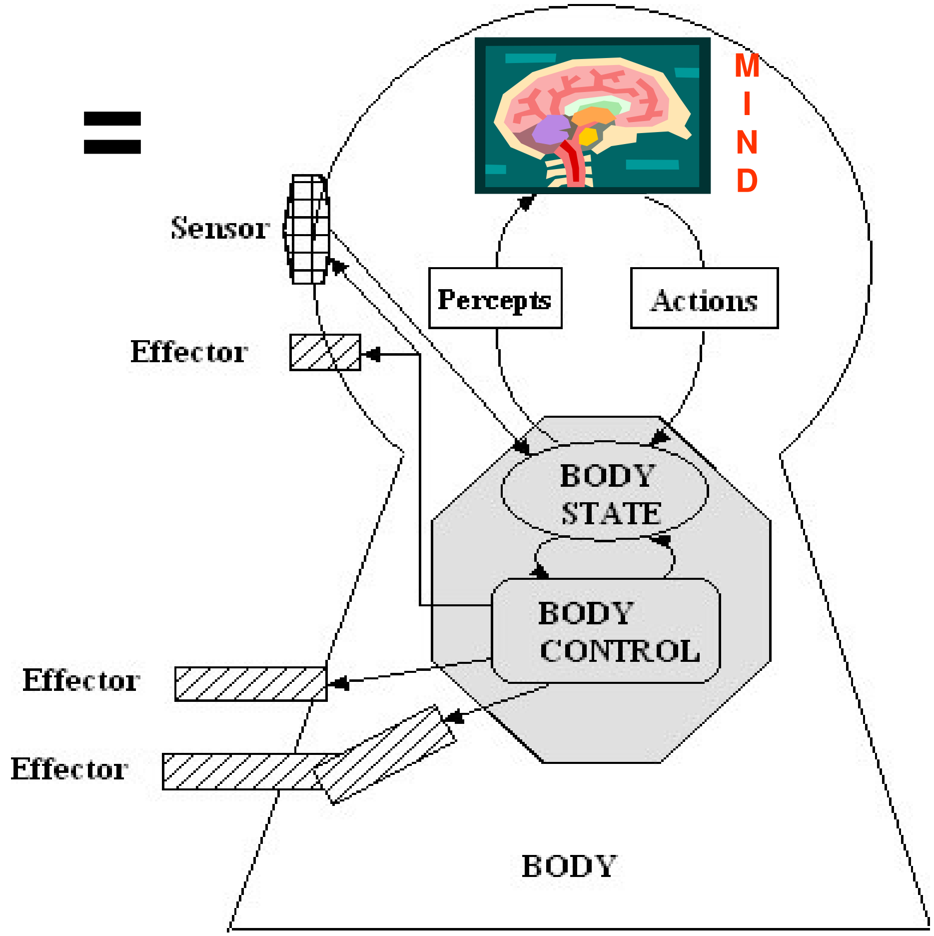
Social/interactive behaviour

- To acquire/share information (e.g. nice restaurants in Bertinoro?)
- To acquire/concede/share resources and services (e.g. could you give me a lift?)
- To “join forces” with other agents (e.g. shall we share a taxi?)
- To provide/get explanations (e.g. you will need sun cream because it is going to be very sunny in Bertinoro)

Agents' architecture

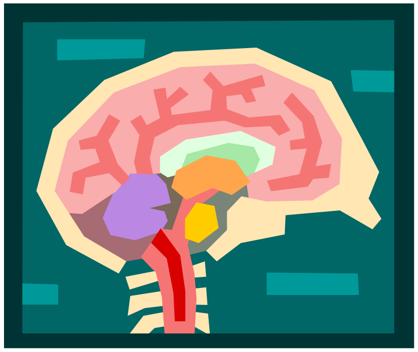


=



- Mind-body:
  - co-routines, i.e. concurrent thinking and action;
  - interruptibility.

Agents' mind



- Data structures ("mental" state)
- Control structure (life "cycle")
- I/O (sensing and acting, including social/communicative behaviour)

Logic-based agents

Logic for representing

- the agent’s mental state
- the agent control (cycle)
- the agent social (communicative) behaviour

Benefits:

- Declarative specification
- Formal specification and verification of properties

Logic programming-based agents

- (various extensions of) LP for
  - designing logic-based agent systems with a clear operational and proof-theoretic counterpart which paves the way to their implementation
- (various extensions of) LP to bridge the gap between
  - theory (high level specification) and
  - practice (execution model) of agents
- Note: the operational specification of many logic-based agent models is grounded in LP

Examples of LP-based MAS

- AgentSpeak(L): PRS (BDI) via Horn-clause logic programs
- IMPACT: agentification of legacy code
- 3APL and 2APL: imperative and declarative model
- KS-agents: IFF abductive proof procedure underlying the observe-think-act cycle of agents
- ALIAS: KM abductive proof procedure for distributed problem solving
- MINERVA: LP-based agents for belief revision-evolution
- DALI: event-based LPs for reactive and proactive behaviour
- AAA: ASP, recovering from unexpected observations
- KGP agents: abductive, constraint, preference-based LP

KS-agents

- Abductive Logic Programs (LPs) to represent agents beliefs, possible actions, observations
- LP queries to represent goals
- Cycle as (interactive) execution of abductive proof procedure - IFF

•Kowalski, Sadri, From Logic Programming Towards Multi-Agent Systems. AMAI 1999
•Fung, Kowalski, *The IFF proof procedure for abductive logic programming*. Journal of Logic Programming 1997.

Abductive LP (and Agents)

LOGIC PROGRAM : ***P***

have(X,Y) ← buy (X,Y).

have(X,Y) ← steal (X,Y).

ABDUCIBLES (HYPOTHESES): ***A***

buy, steal (actions)

no-money (observable)

INTEGRITY CONSTRAINTS : ***IC***

no-money ∧ buy(X,Y) ⇒ **false.**

alarm(T) ⇒ run(T+1).

Semantics of abductive LPs

Given an abductive logic program <P,A,IC> and a ***query*** Q, Δ is an ***abductive explanation*** for Q iff

1. $\Delta \subseteq A$
2. $P \cup \Delta$ "entails" Q (Q is "provable" from $P \cup \Delta$)
3. $P \cup \Delta$ "satisfies" IC (e.g. IC is "provable" from $P \cup \Delta$)

E.g. Q=have(ft,dinner):

Δ_1 = {buy(ft,dinner)} Δ_2 = {steal(ft,dinner)}

Q=have(ft,dinner) **and** no-money:

Δ_1 not ok (Δ_2 ok)

Agents as abductive LPs

- *LOGIC PROGRAMS* represent ***beliefs***

  e.g. have(X,Y) ←buy (X,Y).

- *ABDUCIBLES* represent ***observations*** and ***actions***

  e.g. honest, buy, steal

- *INTEGRITY CONSTRAINTS* represent

  - ***prohibitions*** (e.g. example)
  - ***condition-action*** *and* ***commitment rules*** (e.g. example)
  - ***obligations***

    e.g. request (A,B,X) ∧ have(X) ⇒give(B,A,X).

- *QUERIES* include ***goals*** and ***observations***

Agent behaviour via abductive proof procedure: IFF, CIFF, SCIFF...

QUERY ***EXPLANATION***

OBSERVATIONS	GOALS	ACTIONS
	have(ft,dinner)	
	[steal(ft,dinner)**] or** **[**buy(ft,dinner) **and** **[if** no-money **then false]]**	
no-money		
	[steal(ft,dinner)**] or false**	
		steal(ft,dinner)

IFF proof procedure: example

P: $p \leftarrow a$ $q \leftarrow b$ $r \leftarrow \neg c$

A: a,b,c,d **IC**: $b \wedge r \Rightarrow d$ **Q**: $p \wedge q$

---

1. $p \wedge q \wedge IC$ unfolding p:
2. $a \wedge q \wedge IC$ unfolding q:
3. $a \wedge b \wedge IC$ propagating with b:
4. $a \wedge b \wedge IC \wedge (r \Rightarrow d)$ unfolding r :
5. $a \wedge b \wedge IC \wedge (\neg c \Rightarrow d)$ negation elimination:
6. $a \wedge b \wedge IC \wedge (c \vee d)$ splitting:
7. $(a \wedge b \wedge IC \wedge c) \vee (a \wedge b \wedge IC \wedge d)$

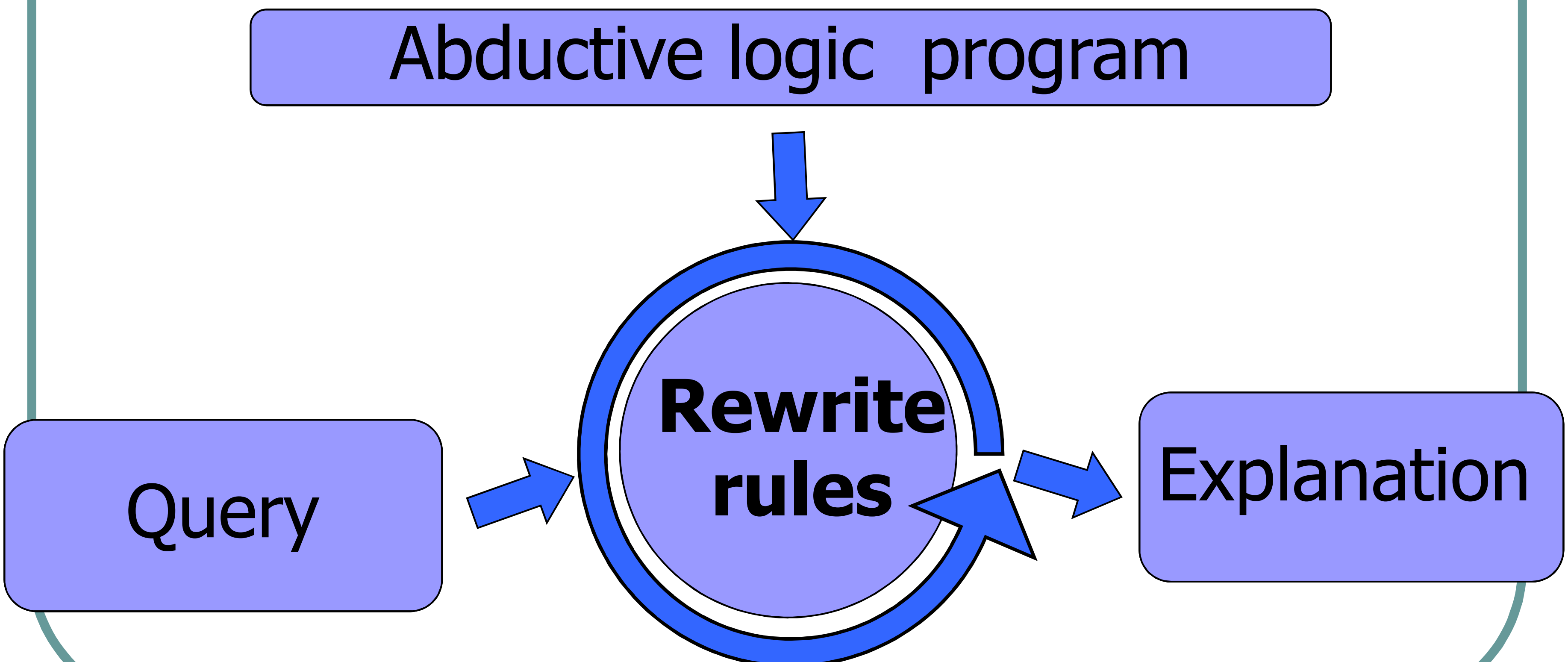
Rewrite Rules

Two answers: {a,b,c} {a,b,d}

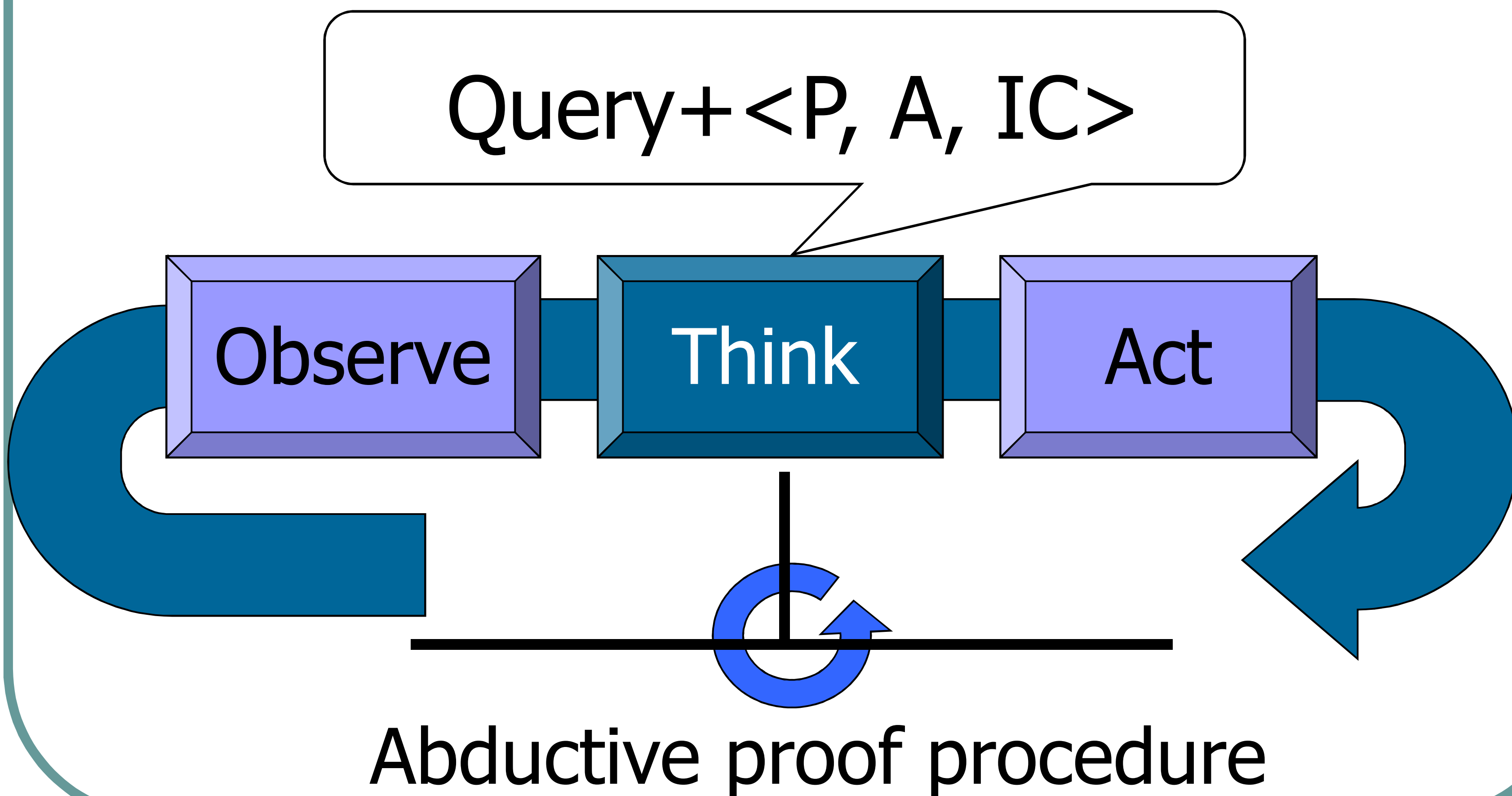
CIFF also deals with constraints as in CLP

Execution of abductive proof procedures

- Static (off-line queries/explanations)
- ***Interactive*** and ***resource-bounded*** (on-line queries/explanations)



(Abductive) Agent behaviour: cycle



Part I (revisited)

Agents

- General Introduction
- Logic- and LP-based approaches
- KGP agents

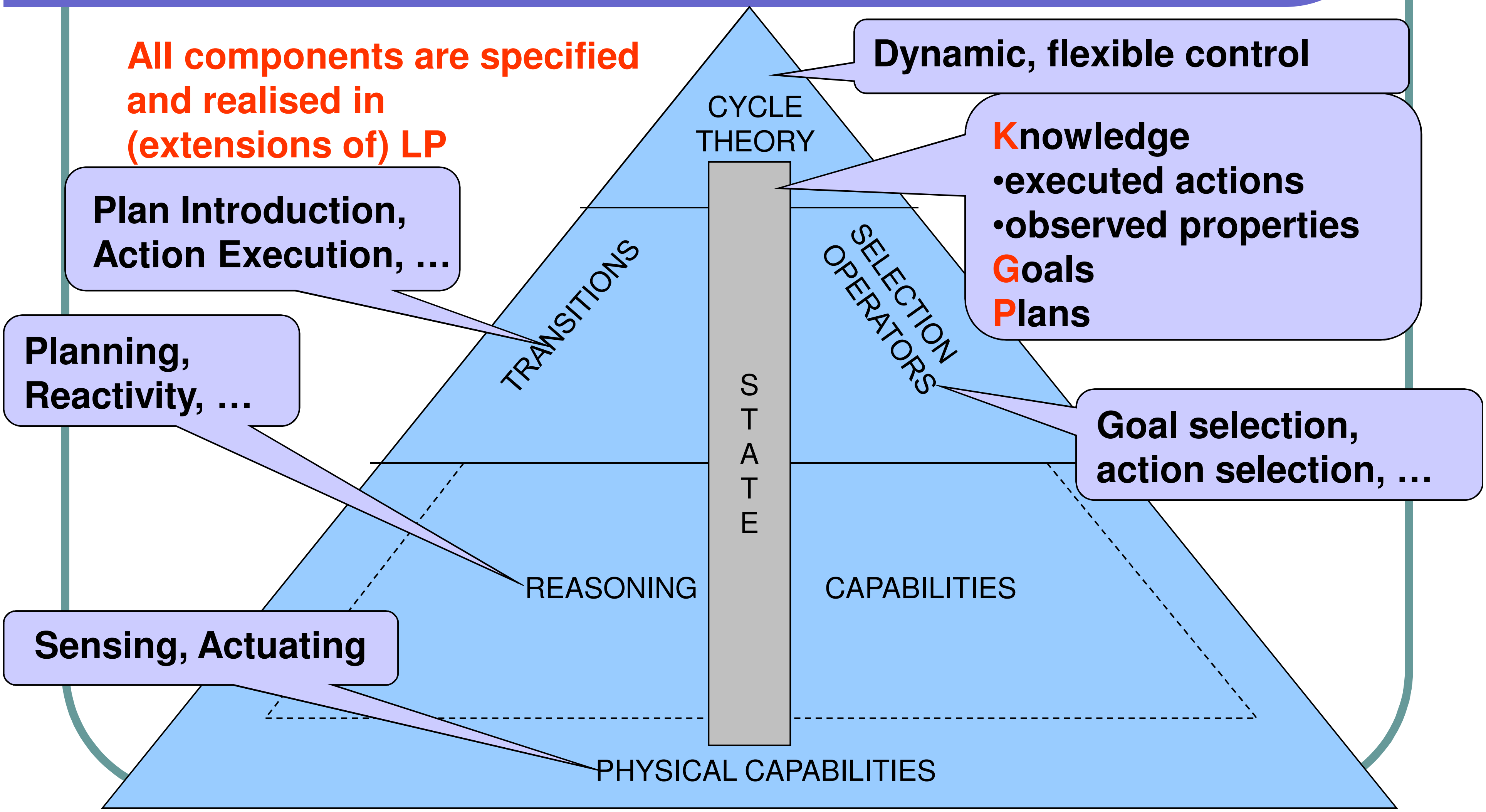
•A. Kakas, P. Mancarella, F. Sadri, K. Stathis, F. Toni, *Computational Logic Foundations of KGP Agents*, Journal of Artificial Intelligence Research, Volume 33, pages 285-348, 2008

KGP: motivations and main features

- KGP is a general-purpose, highly modular architecture for agents with
  - an abstract model ("declarative" semantics)
  - a computational model (operational semantics)
  - a prototype implementation (PROSOCS)
- KGP focuses on the needs of agents in a dynamic and open setting
- KGP integrates various aspects of agency: pro-activeness, autonomy, reactivity, social ability (communication)
- The computational model of KGP extends and integrates various existing LP theories and proof procedures

KGP agents' mind: an overview

All components are specified and realised in (extensions of) LP



KGP agents: an overview (more)

- Agent behaviour is given by
  - a sequence of state-changing transitions ("calling" no/one/many capabilities)
  - with inputs provided by selection operators
  - as "decided" by the cycle theory
- For example

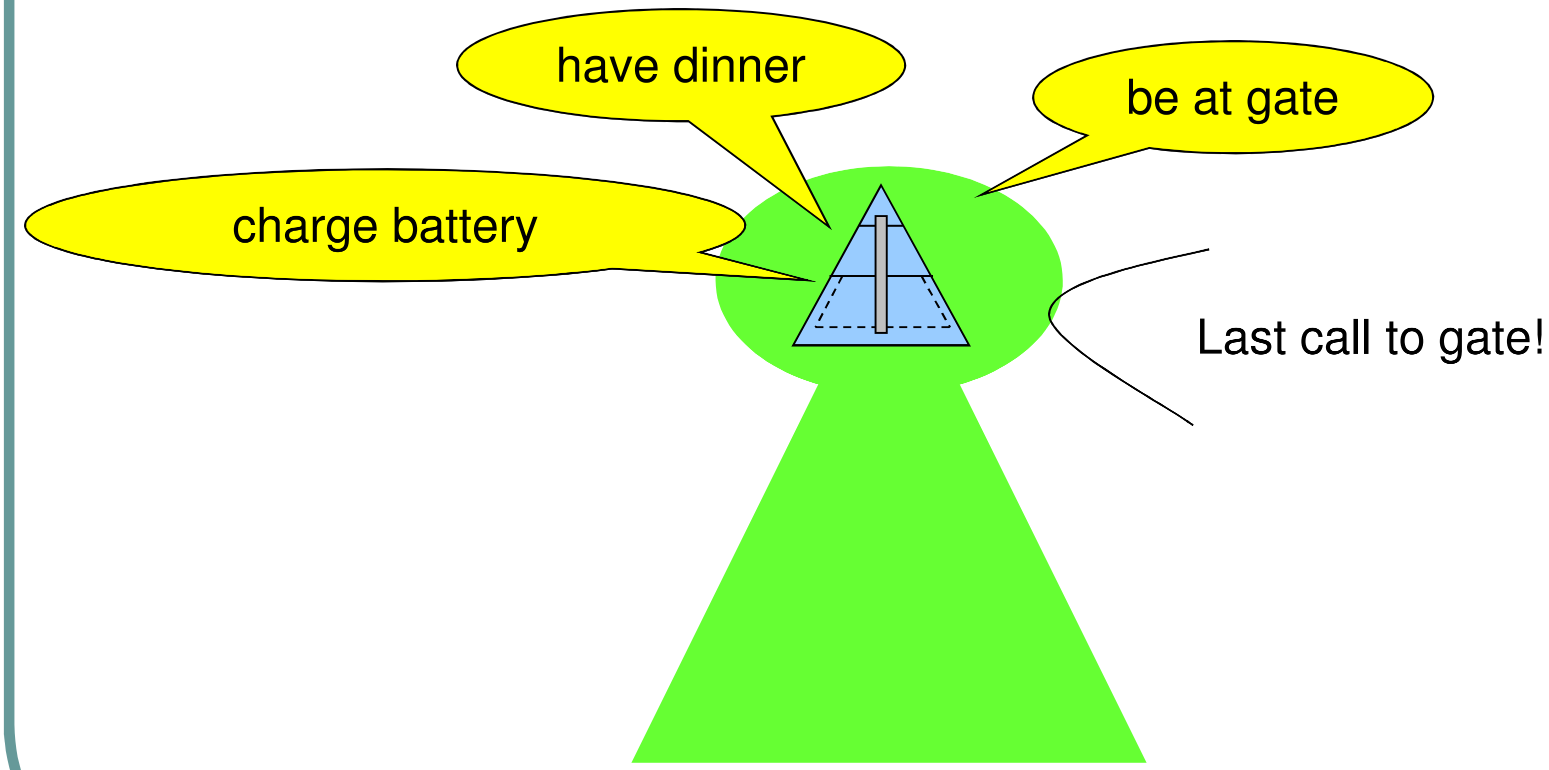
plan for a chosen goal

execute some chosen actions in the plan

observe the environment (possibly "actively")

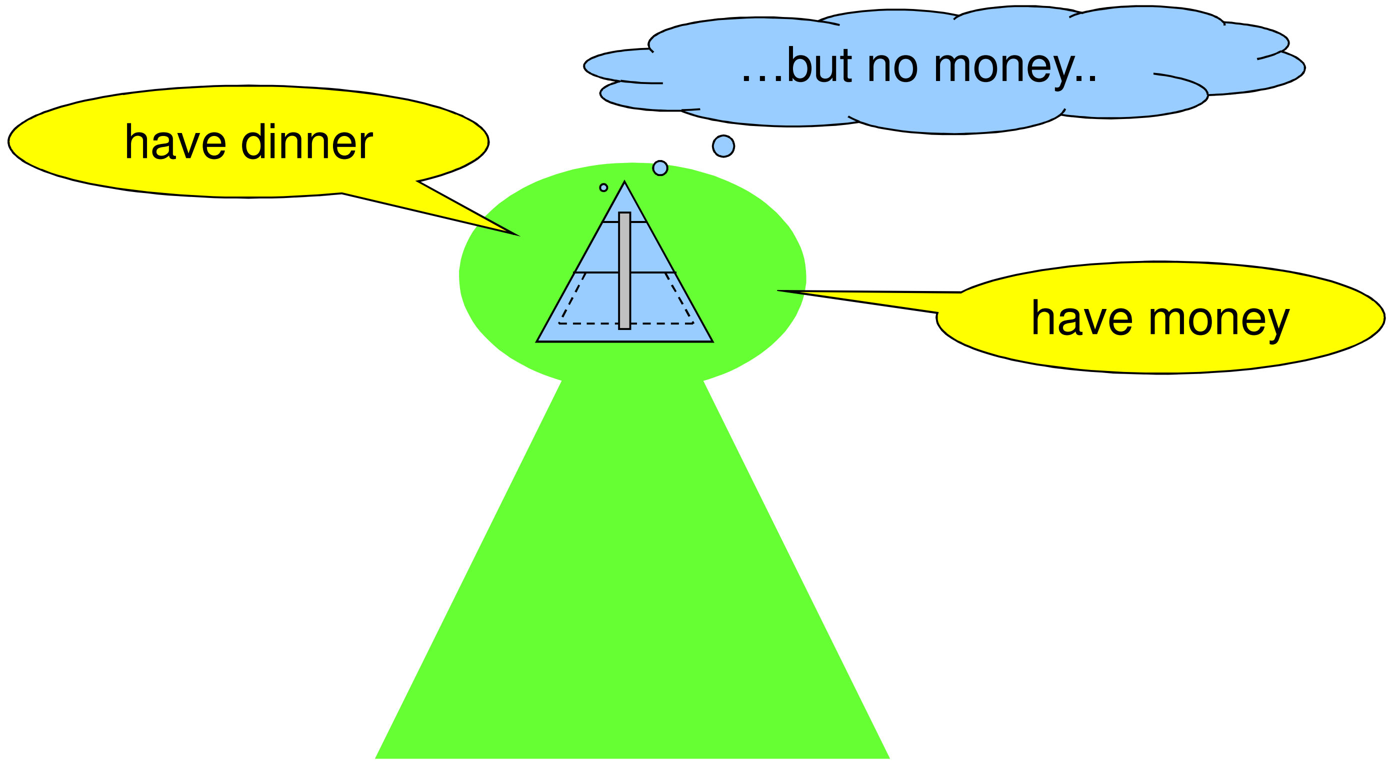
reassess your goals/react to any changes …

Goal decision/re-assessment



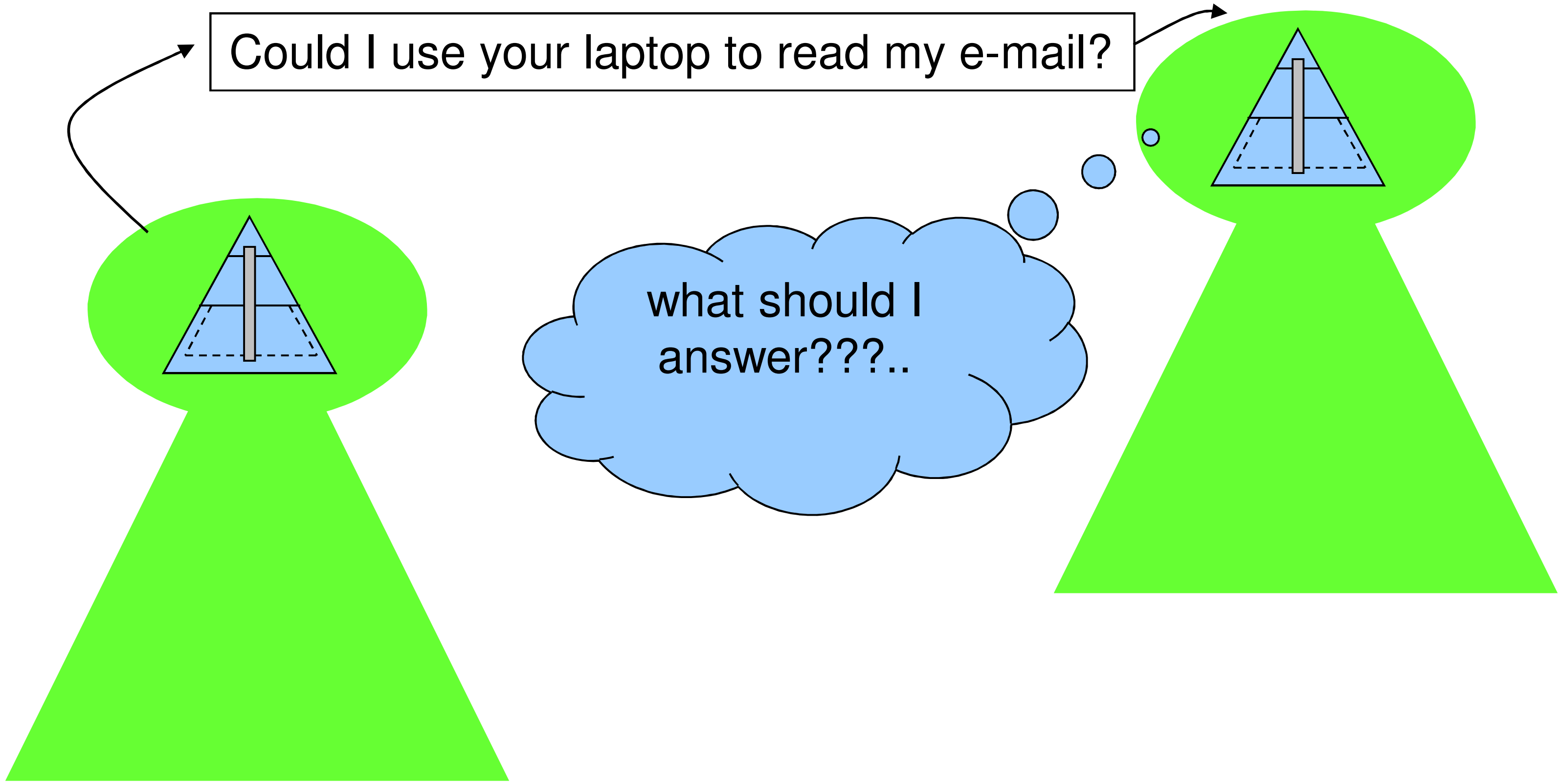
KGP agents adjust their goals to observations

Plan adjustment



KGP agents adjust their plans to circumstances

Reactivity and Interactivity



KGP agents: an overview (more)

- Agent behaviour is given by
  - a sequence of state-changing transitions ("calling" no/one/many capabilities)
  - with inputs provided by selection operators
  - as "decided" by the cycle theory
- Another example

plan for a chosen goal

observe the environment (possibly "actively")

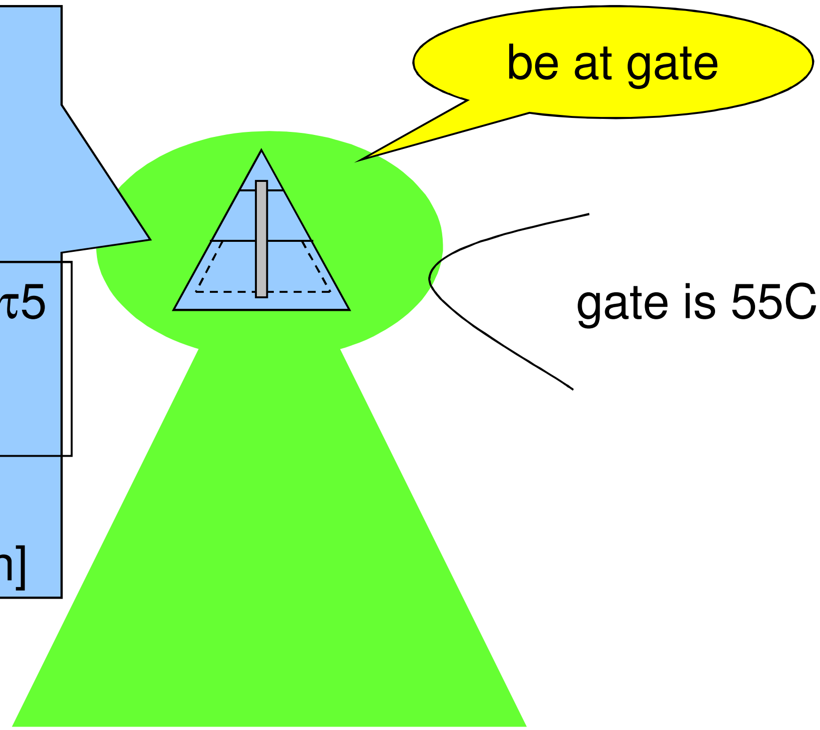
plan a little more

execute some chosen actions in the plan,…

Interleaving planning and observation

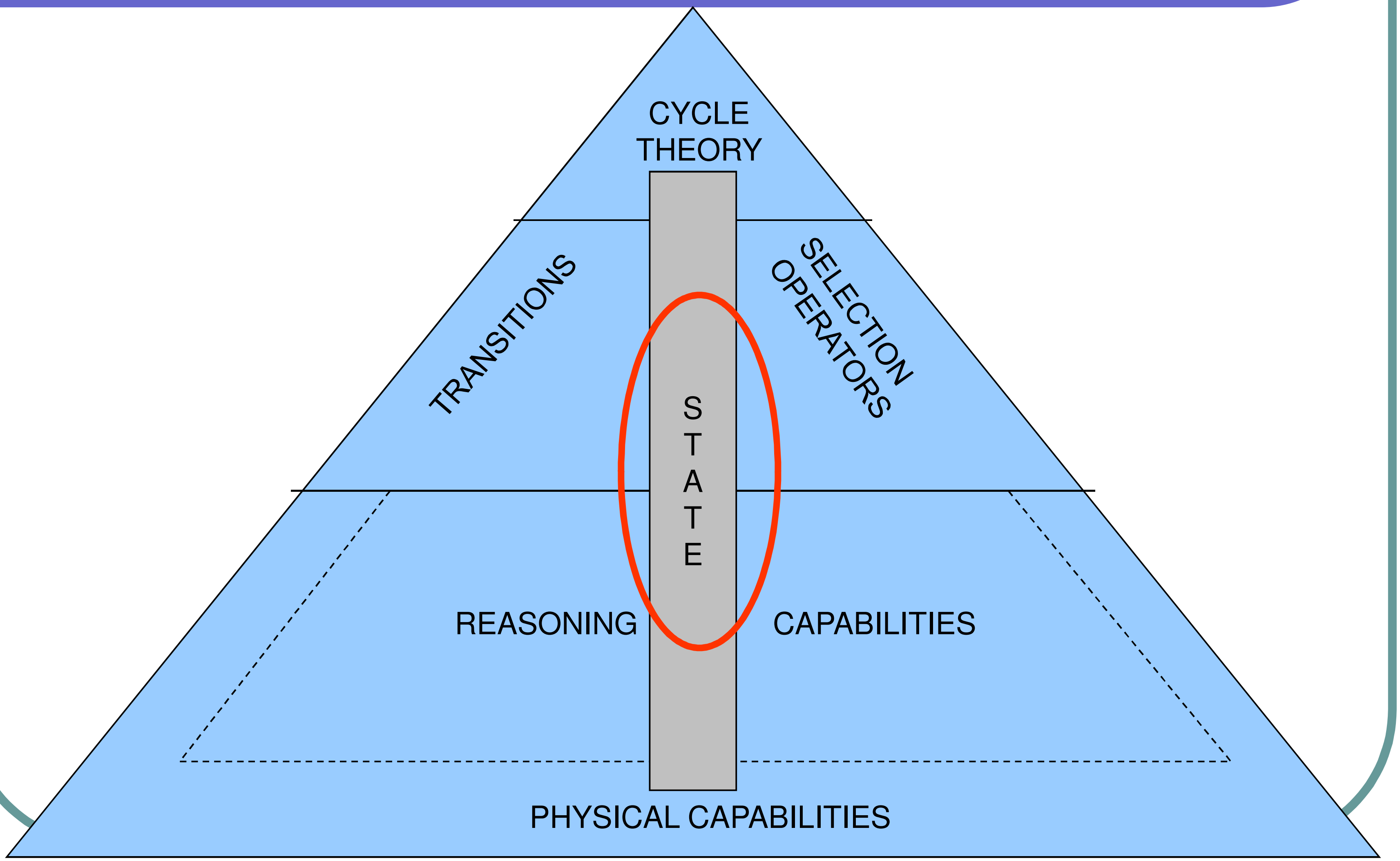
- Go to security check at τ1
- Go to hall at τ2
- Check gate at τ3
- Go from hall to gate at τ4
  - Get transfer to hallB at τ5
  - Walk to gate 55C at τ6
  - τ3<τ5<τ6<d

now<τ1<τ2<τ3<τ4<d
[d=departure time – 20 min]



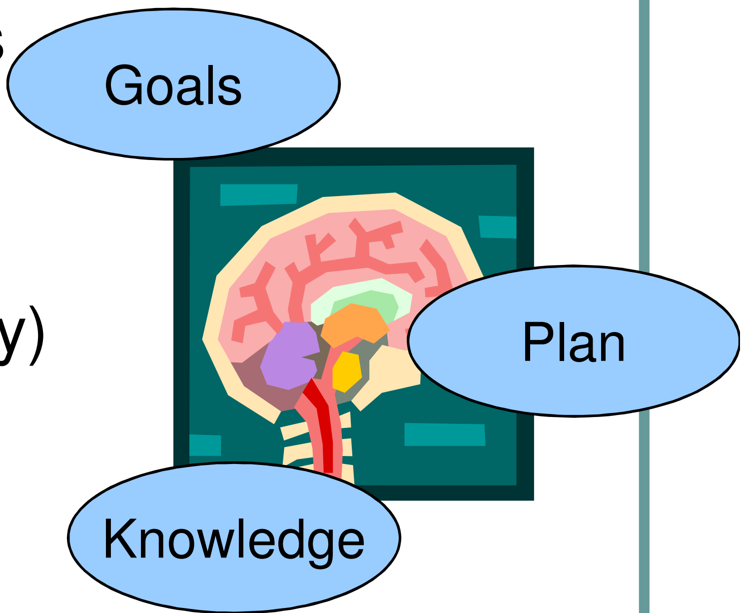
KGP plan “partially”, observe actively, adjust their plans

KGP agents' mind: an overview



Mental state of an agent

- KB$_0$ contains
  - a representation of observations in the environment (sensing capability)
  - a record of execution of actions by the agent (actuating capability)
- Goals is a concrete set of (mental/sensing) goals
- Plan is a concrete set of (physical / communicative / sensing) actions



Mental and sensing goals

- Mental goals: goals whose fluents represent properties the agent itself is able to plan for so that they can be satisfied, but can also be observed.

  e.g. *have(dinner,τ), have(money,3)*

- Sensing goals: goals whose fluents represent properties which are not under the control of the agent and can only be observed by sensing the external environment.

  e.g. *which-gate(t)*, *request_accepted(τ)*

Physical, Communicative, Sensing actions

- "Physical" actions: that the agent performs in order to achieve some specific effect (and typically causing changes in the environment)

e.g. *go(hall,τ)*

- Sensing actions: that the agent performs in order to establish whether some properties (fluents) hold in the environment or not

e.g. *sense(connection_on,τ)*

- Communicative actions: that involve communications with other agents

e.g. request(x, y, give(laptop),τ)

Goals and Actions

- Are assigned an explicit time (implicitly existentially quantified within the state) with associated temporal constraints in the state
- Are organised within a forest of trees structure
  - for ease of revision and to allow continual planning:
    - Top-level goals/actions (the roots of trees)
    - Sub-goals (goals with ancestors in the tree)
    - Actions are leaves (but can be top-level)
- Roots (and their trees) may be reactive or not

Goals/actions concretely

- A goal *G* is a *(timed) fluent* $l[\tau]$

  e.g. $l[\tau]$= *have(dinner,* τ*)*

  $l[\tau]$= ¬*have(money,* τ*)*
- An action is a *(timed)* action literal $a[\tau]$

  e.g. $a[\tau]$= *request(x,y,give(laptop),* τ*)*

  where
  - *have* is a fluent
  - *request* is an action operator
  - τ is the time (a variable implicitly existentially quantified within the state)

A state formally

$S = <KB_0, F, C, \Sigma>$

- F is the forest
- C is the set of temporal constraints
- Σ is a set of equalities (time var = time constant)

Notes:

- The valuation of C always take Σ into account
- Constraint solving capability $\models_{cs}$ "uses" C and Σ:

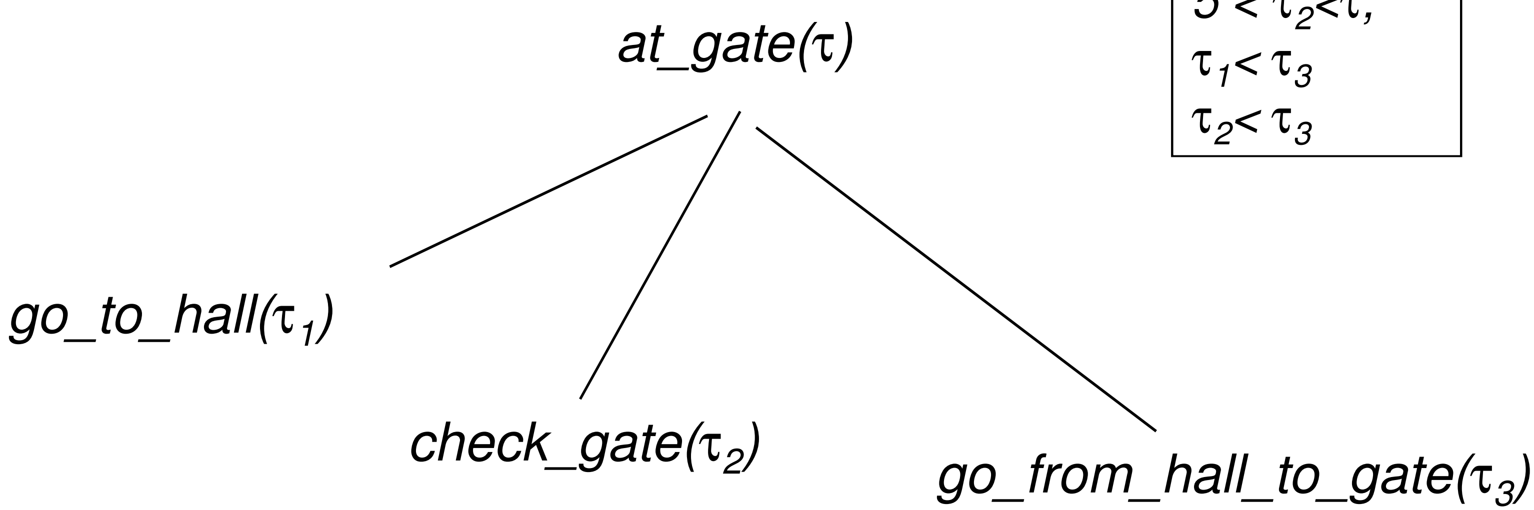
  $S \models_{cs} TC$ *iff* $\models_{\Re} C \wedge \Sigma \wedge TC$

  e.g. C={} and Σ ={}: $S \models_{cs} \tau < 10 \wedge \tau > 3$

  C={} and Σ ={$\tau = 2$}: $S \not\models_{cs} \tau < 10 \wedge \tau > 3$

Example of a tree in the state

$5 < \tau < 10,$
$5 < \tau_1 < \tau,$
$5 < \tau_2 < \tau,$
$\tau_1 < \tau_3$
$\tau_2 < \tau_3$



Temporal constraints (an example)

TC ::= AtomicTC | TC $\wedge$ TC | TC $\vee$ TC | $\neg$TC

AtomicTC ::= Variable RelOp Term

RelOp ::= = | $\leq$ | < |

Term ::= Time_Constant |
Time_Variable |
Term Op Term

Op ::= + | - | * | \

e.g. $\tau < 3 + \tau'$

$\tau < 3 + \tau' \wedge \tau' < 10$

Constraint solving (reasoning)capability:
$\models_{cs} \simeq \models_{\Re}$

KB_0

- KB_0 *revised* as the agent observes the world (via its sensing capability) and executes actions (via its actuating capability)
- KB_0 *is the narrative part* within the KBs underlying and reasoned upon by (most) reasoning capabilities

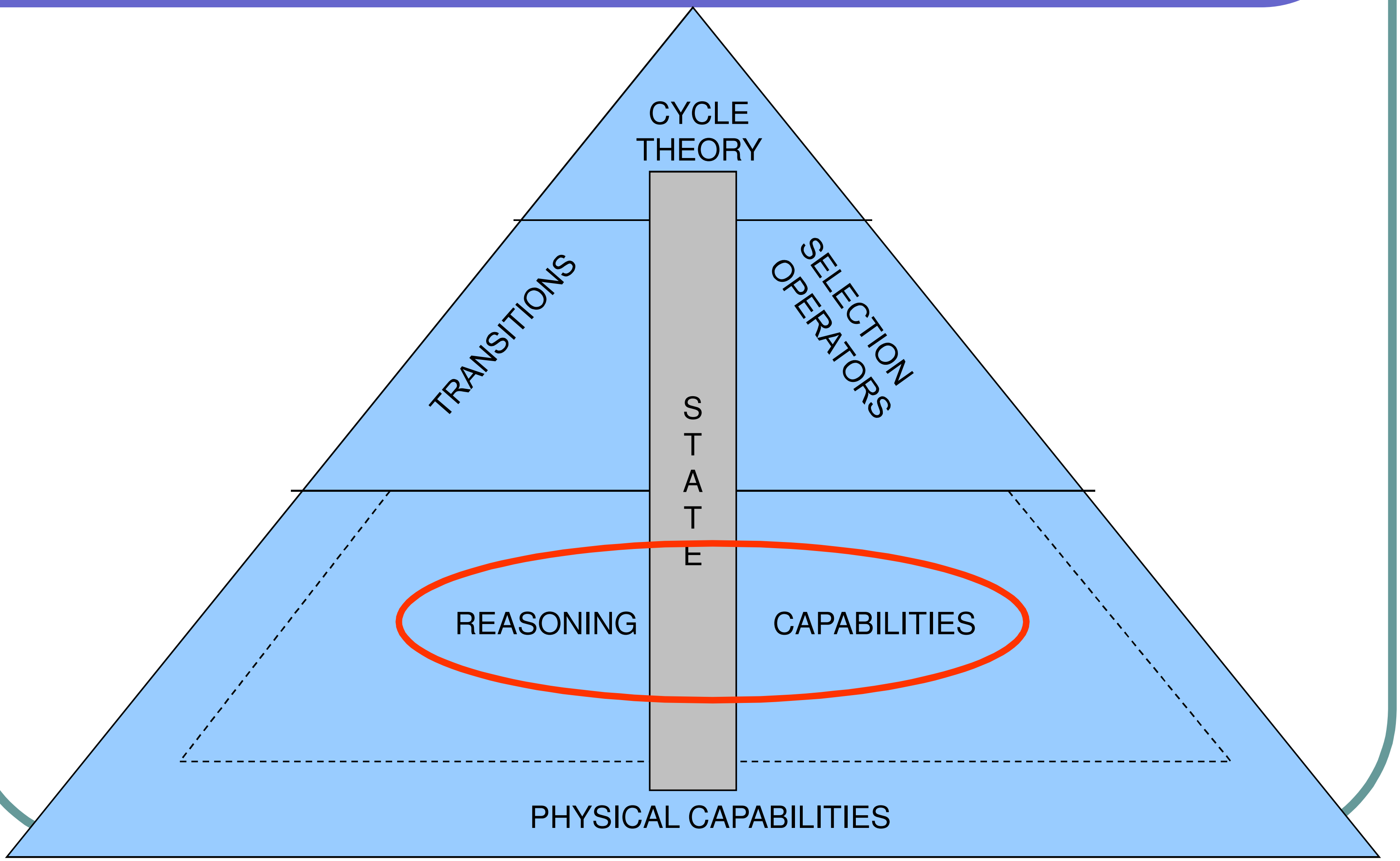
$$KB_{pre}, KB_{eff}, KB_0 \subset KB_{TR} \subset KB_{plan} \subseteq KB_{react}$$

$$KB_{TR} \subset KB_{GD}$$

KB_0

- KB_0 of agent *x* records:
  - *actions (timed action operator)* which have been executed by *x* (+ time of execution $\tau = \mathrm{t}$ in Σ)
    executed(a[t]) (where *t* is ground)
  - *actions* which have been executed by agents *y* other than *x* (+ time of execution by *y, if known,* + time of observation by *x*)
    observed(y,a[t'], t) (where *t* , *t'* are ground)
  - properties (*fluent literals*) observed (+ time of observation $\tau = \mathrm{t}$ in Σ)
    observed(l[t]) (where *t* is ground)

KGP agents' mind: an overview



Reasoning capabilities

- Planning, $\models^{now}_{plan}$
  - *generates partial plans (sets of actions and goals to be added to the State) for given sets of goals in the State*
- Reactivity, $\models^{now}_{react}$
  - *reacts to perceived changes in the environment (KB_0) by generating goals and actions to be added to the State*
- Temporal reasoning, $\models_{TR}$
  - *reasons about the perceived environment (KB_0), and makes predictions about properties holding in the environment*
- Identification of preconditions/ effects $\models_{pre}$, $\models_{eff}$
  - *for actions (before/after they are executed)*
- Goal decision, $\models^{now}_{GD}$
  - *revises the top-most level goals of the agent, taking into account its preferences & the perceived changes in the environment (KB_0)*

Reasoning capabilities: I/O view

- Planning:

$$S,G_1,\ldots,G_n \models^{now}_{plan} (As_1,Gs_1,TC_1),\ldots,(As_n, Gs_n ,TC_n)$$

- Reactivity:

$$S \models^{now}_{react} (As,Gs,TC)$$

- Temporal reasoning:

$$S \models_{TR} l[\tau] \wedge TC$$

- Goal decision:

$$S \models^{now}_{GD} (Gs,TC)$$

- Identification of preconditions, effects :

$$S, A \models_{pre} Gs$$

$$S, A \models_{eff} Gs$$

Reasoning capabilities and LP

- Constraint solving
  - constraint programming $\models_{\Re}$
- Identification of preconditions/effects
  - logic programming, $\models_{LP}$
- Temporal reasoning
  - constraint logic programming, $\models_{LP(\Re)}$
- Planning
- Reactivity
  - abductive (constraint) logic programming, $\models_{LP}$
- Goal decision
  - (constraint) logic programming with priorities, $\models_{pr}$

Event calculus

Underlying (C)LP-based semantics

- The KGP model is parametric on:
  - $\models_{LP}$, some semantics for logic programs with negation
  - $\models_{pr}$, some semantics for logic programs with priorities (and negation)
  - $\models_{\Re}$, some constraint satisfaction tool
- The operational counterpart for KGP assumes:
  - $\models_{LP}$= 3-valued completion semantics
  - $\models_{pr}$= argumentation-based semantics for LPwNF (a concrete framework for logic programming with priorities)

Core Event Calculus: domain independent part

$$holds\text{-}at(F,T_2) \leftarrow happens(Op,T_1),\ T_1 < T_2,\ initiates(Op,T_1,F),\ not\ clipped(T_1,F,T_2)$$

$$holds\text{-}at(F,T) \leftarrow initially(F),\ 0 \leq T,\ not\ clipped(0,F,T)$$

$$clipped(T_1,F,T_2) \leftarrow happens(Op,T),\ terminates(Op,T,F),\ T_1 \leq T < T_2$$

Core Event Calculus: domain independent part (cntd)

holds-at($\neg$F,T_2) $\leftarrow$ happens(Op,T_1), $T_1 < T_2$,
terminates(Op, T_1,F),
not declipped(T_1, F, T_2)

holds-at($\neg$F,T) $\leftarrow$ initially($\neg$F), $0 \leq T$,
not declipped(0,F,T)

declipped(T_1,F,T_2) $\leftarrow$ happens(Op,T),
initiates(Op,T,F), $T_1 \leq T < T_2$

Core Event Calculus: domain dependent part - example

initially(have_money)

precondition(buy_dinner,T,have_money)

initiates(get_cash,T,have_money)

terminates(buy_dinner,T,have_money)

Core Event Calculus: bridge rules (connecting to KB_0)

holds-at(F,T_2) ← observed(F,T_1), $T_1 \leq T_2$,

not clipped(T_1,F,T_2)

holds-at(¬ F,T_2) ← observed(¬ F,T_1), $T_1 \leq T_2$,

not declipped(T_1,F,T_2)

happens(Op,T) ← executed(Op,T)

happens(Op,T) ← observed(_,Op[T],_)

clipped(T1, F, T2) ← observed(¬F, T), T1 ≤ T < T2

declipped(T1, F, T2) ← observed(F, T), T1 ≤ T < T

Temporal reasoning, identification of preconditions and effects

- KB_{TR} = *core EC*
- $S \vDash_{TR} l[\tau] \wedge TC$ *iff* $KB_{TR} \vDash_{LP(\Re)} holds\_at(l,\tau) \wedge TC$

- KB_{pre} = *rules for* **precondition** *in core EC*
- $S, a[\tau] \vDash_{pre} Cs$ *iff*

  $Cs = \{l[\tau] \mid KB_{pre} \vDash_{LP} precondition(a, l)\}$

- KB_{eff} = *rules for* **init./term.** *in core EC*
- $S, a[\tau] \vDash_{eff} G$ *iff*

  $KB_{eff} \vDash_{LP} initiates(a, f)$ *and* $G=f$

  $KB_{eff} \vDash_{LP} terminates(a, f)$ *and* $G=\neg f$

KB_{plan} : Abductive EC

KB_{plan} = < P_{plan}, A_{plan}, I_{plan} > abductive logic program

P_{plan} core EC +

happens(Op,T) ← assume_happens(Op,T)

holds_at(F, T) ← assume_holds(F, T)

A_{plan}: assume_happens(Op,T)

assume_holds(Op,T)

I_{plan} :

holds_at(F,T) and holds_at(¬F,T) ⇒ false

assume_happens(Op,T) , precondition(Op,P) ⇒ holds_at(P,T)

assume_happens(Op, T), not executed(Op, T), time_now(T1)⇒ T > T1

Planning

- $S, G \models^{now}_{\mathbf{plan}}$ As,Gs, TC

iff "assume_happens(As)"+ "assume_holds(Gs)" +TC

is an abductive explanation

- wrt KB_{plan} ∪ {time_now(now)} ∪ "assume_happens(F)"+ "assume_holds(F)"
- for query "holds_at(G)"

- $S, G \models^{now}_{\mathbf{plan}} \bot$

iff no such abductive explanation exists

KB_{react}

KB_{react} = KB_{plan} + Reactive Constraints

(as additional integrity constraints)

- **Reactive constraints**:

  Triggers, Other-Conditions ⇒ Reaction and Constraints

  - *Triggers* is a non-empty conjunction of items of the form *observed* (l[T], T'), *observed* (c, a[T], T'), *happens(a[T],T')*
  - *Other-conditions* is a conjunction of any of the following:
    - *holds-at* (l,T'), where l[T'] is a timed fluent literal
    - *happens* (a,T'), where l[T'] is a timed action operator
    - temporal constraints
  - *Reaction* is either a timed fluent literal or a timed action operator

Reactive Constraints in KB_{react}

Triggers, Other-Conditions ⇒ Reaction and Constraints

Observations

Executed actions constraints

Planned actions

holds-at(l,T)

happens(a,T)

temporal constraints

timed fluent literal

timed action op.

temporal

Examples of reactive constraints

Interaction policy:

observed(C, tell(C,a,request(R),D,T1),T) $\wedge$
holds-at(have(R),T1)) $\wedge$ *not holds-at(need(R),T1))* $\Rightarrow$
assume-happens(tell(a,C,accept(request(R)),D,T1),T2) $\wedge$
T+5>T2>T

Condition-action rule:

observed(alarm-sound(Room),T) $\wedge$ *holds-at(in(Room),T))* $\Rightarrow$
assume-happens(leave(Room),T1) $\wedge$
$T1 \leq T + 2$

Reactivity

- $S, G \models^{now}_{\mathbf{react}}$ As,Gs, TC

if "assume_happens(As)"+ "assume_holds(Gs)" +TC

is an abductive explanation

  - wrt $KB_{react} \cup$ {time_now(now)} $\cup$ "assume_happens(F)"+ "assume_holds(F)"
  - for the **empty query**

- $S, G \models^{now}_{\mathbf{react}} \perp$

if no such abductive explanation exists

KB_{gd} (Example)

gd(dinner): have_dinner(T) $\leftarrow$
holds_at(hungry, T)

gd(gate): at_gate(T) $\leftarrow$
holds_at(boarding, T)

incompatible(have_dinner(T),at_gate(T))

typeof(dinner,optional)

typeof(gate,required)

more_urgent_wrt_type(required, optional)

gd_pref(X,Y):gd(X)<gd(Y) $\leftarrow$
type_of(X,TX), type_of(Y,TY),
more_urgent_wrt_type(TY,TX)

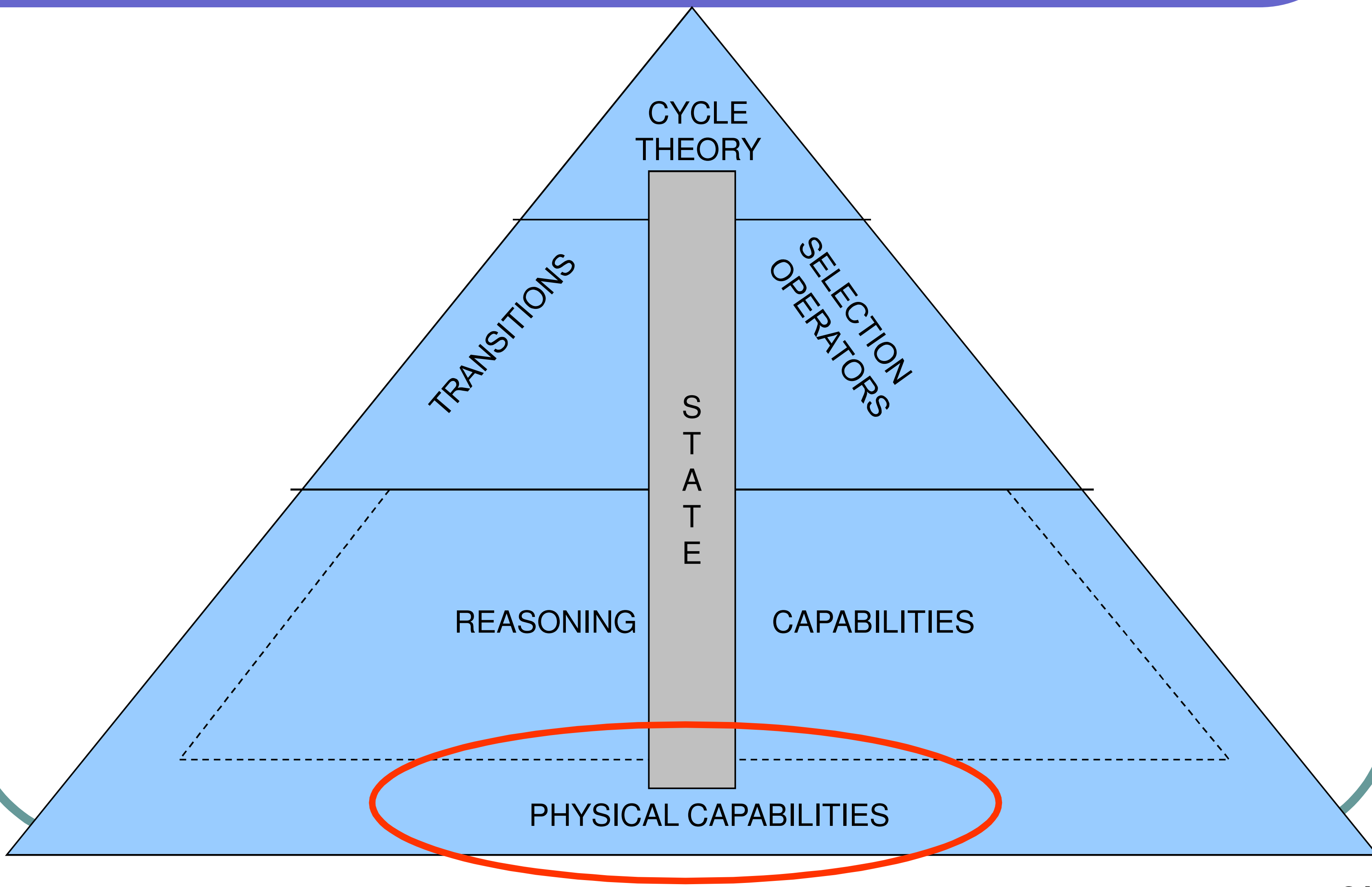
Goal decision (Example)

- If $S \vDash_{\mathbf{TR}}$ holds_at(hungry, τ) then
  $S \vDash^{now}_{\mathbf{GD}}$ have_dinner(τ)

- If $S \vDash_{\mathbf{TR}}$ holds_at(boarding, τ) then
  $S \vDash^{now}_{\mathbf{GD}}$ at_gate(τ)

- If $S \vDash_{\mathbf{TR}}$ holds_at(hungry, τ) $\wedge$holds_at(boarding, τ)
  then
  $S \vDash^{now}_{\mathbf{GD}}$ at_gate(τ)

KGP agents' mind: an overview

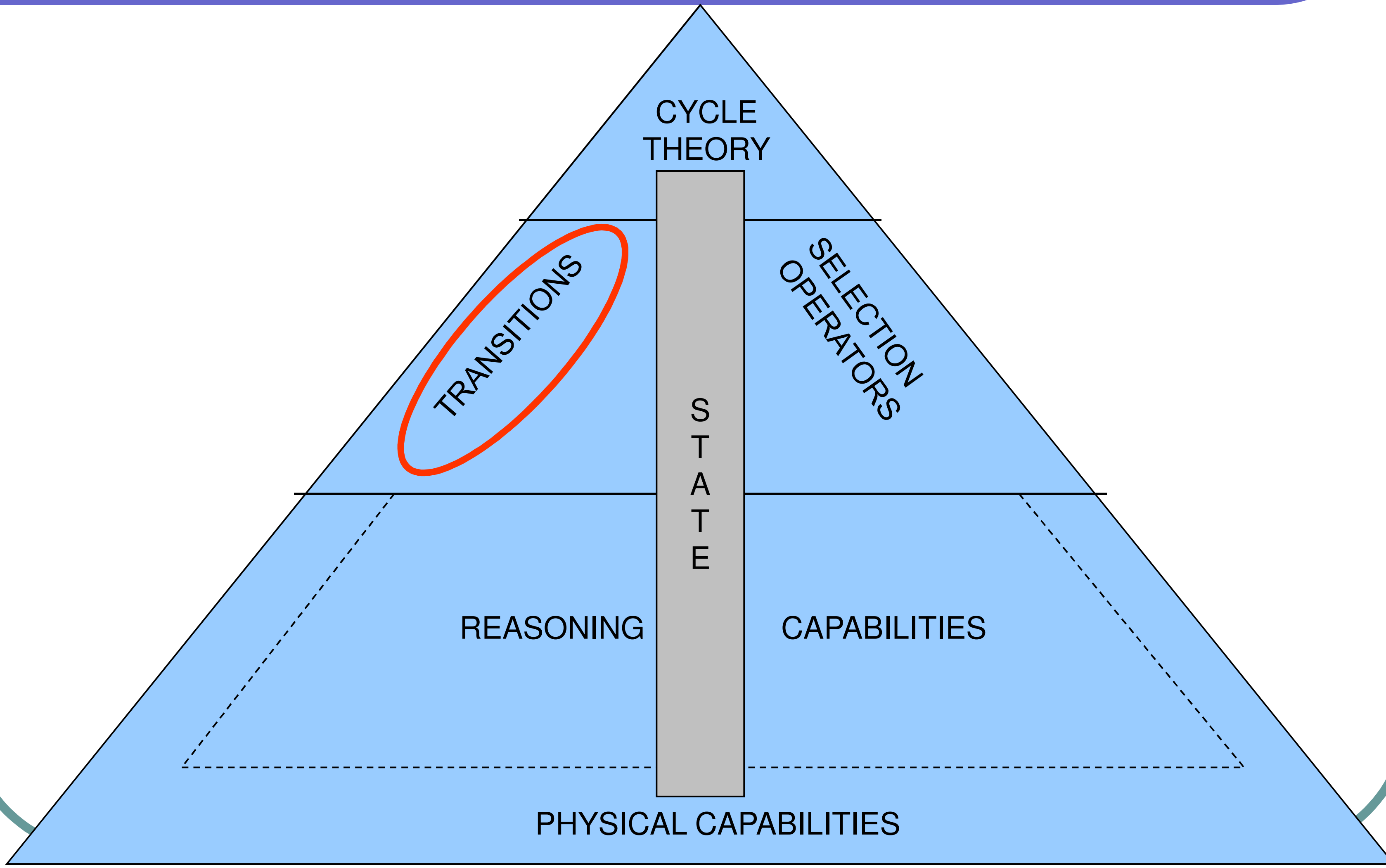


Physical capabilities

Link the agent to its environment

- sensing(L, now) = L'
  - L is a set of fluent literals *l or c:a (another agent has performed action a)*
  - L' is a set of observations (l:true; l:false; c:a[t])
- actuating(As, now) = As'
  - As is a set of actions to be executed at time now
  - As' is the subset of As that the body actually managed to execute

KGP agents' mind: an overview



Transitions: general idea

- Transitions are of the form

$$(T)\frac{S = <KB_0, F, C, \Sigma>, X}{S = <KB_0', F', C', \Sigma'>,} now$$

  - T is the name of the transition
  - X is some additional (possibly empty) input
  - now is the time of application of the transition

- Transitions typically call some capabilities
- Inputs computed by selection operators, also calling capabilities

Transitions in a nutshell

- Goal Introduction (GI), introduces new goals into the state
- Plan Introduction (PI), performs (partial) planning for the *input* goals and extends the state accordingly
- Reactivity (RE), extends the state by means of generated reactions
- Action Execution (AE) executes (into environment) all actions in the *input* set
- State Revision (SR) revises the forest (goals/actions)

Transitions in a nutshell (cntd)

- Passive Observation Introduction (POI) senses the environment

  (about whatever it has to offer)
- Active Observation Introduction (AOI) senses the environment

  (about the *input* set of fluents, effects of some executed actions)
- Sensing Introduction (SI), introduces new sensing actions in the state

  (for sensing the preconditions, given in *input*, of some existing actions)

Transitions and capabilities

- Goal Introduction (GI), calling $\models^{now}_{GD}$ and $\models_{cs}$
- Plan Introduction (PI), taking a *set of goals* in *input* and calling $\models^{now}_{plan}$
- Reactivity (RE), calling $\models^{now}_{react}$ and $\models_{cs}$
- Sensing Introduction (SI), taking a *set of preconditions of actions* in *input* and calling $\models^{now}_{pre}$
- Passive Observation Introduction (POI), calling *sensing*
- Active Observation Introduction (AOI), taking a set of *fluent literals* in *input* and calling *sensing*
- Action Execution (AE), taking a *set of actions* in *input* and calling *sensing* and *actuating*
- State Revision (SR), calling $\models_{TR}$ and $\models_{cs}$

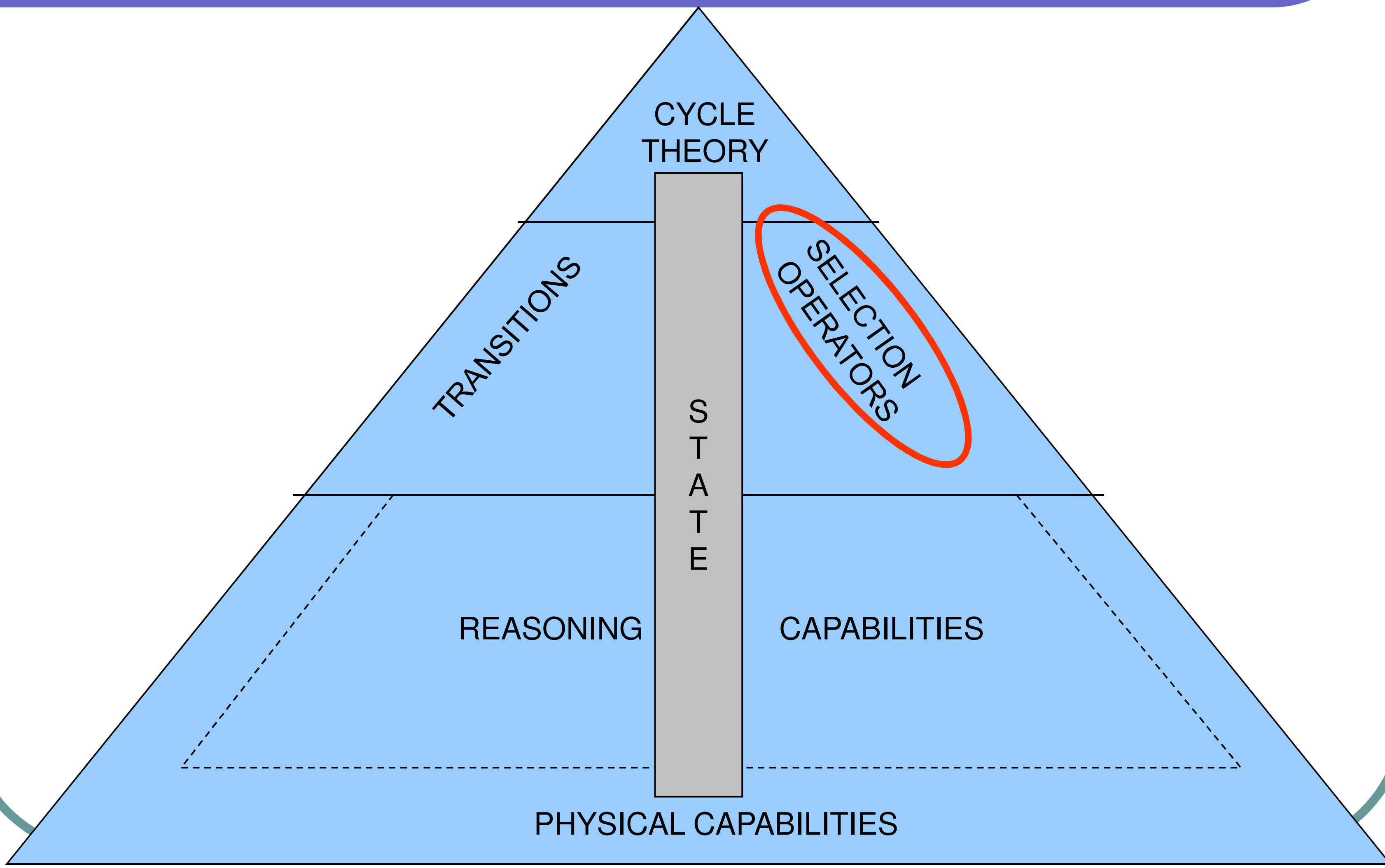
An example: State Revision (informally)

$$(SR)\ \frac{S = \langle KB_0, F, C, \Sigma \rangle, \{\}}{S = \langle KB_0, F', C, \Sigma \rangle,}\ \text{now}$$

F' is the biggest subset of F consisting of all goals/actions G in F that

- Are not timed-out
- Have not been executed (if actions)/achieved (if goals)
- Their siblings (in trees) have not been removed because timed-out
- Their ancestors have not been removed
- They are not sensing actions for preconditions of actions that have been removed

KGP agents' mind: an overview



Selection operators

- Compute inputs to transitions
- Help cycle theories (control, see later) determine next transition
- 4 core selection operators
  - Action selection (to execute, for AE)
  - Goal selection (to plan for, for PI)
  - Fluent selection (to sense, for AOI, effects of executed actions)
  - Precondition selection (to plan for sensing, for SI, preconditions of actions under consideration for execution),
- E.g. execute “urgent” actions

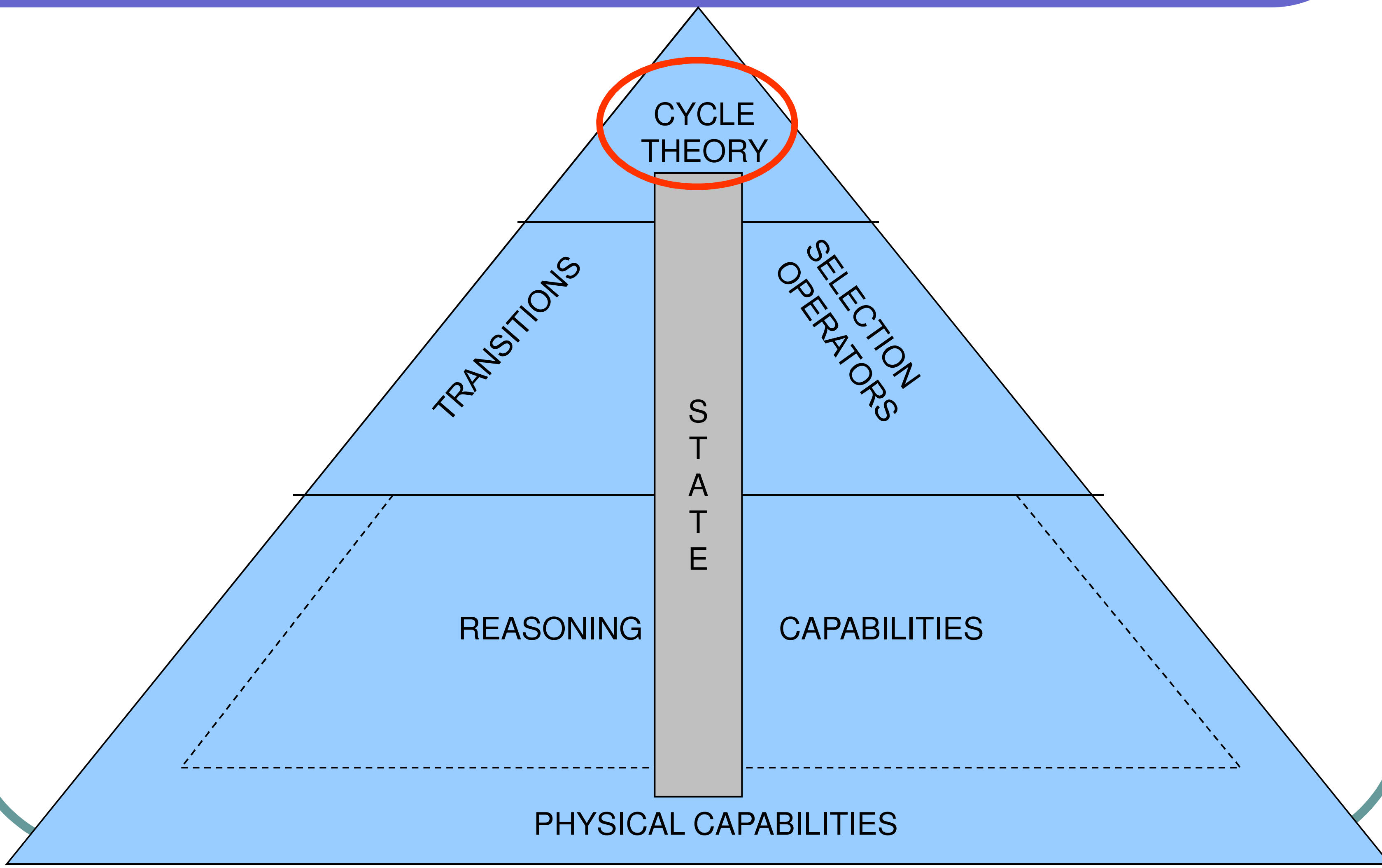
Precondition Selection (informally)

Selects all pairs (*C,A*) *of (timed) preconditions C and actions* $A \in$ *nodes*(*F*) *such that:*

1. *C is a precondition of A* ($\models_{\mathbf{pre}}$) ,
2. *C is not known to be true in S now* ($\models_{\mathbf{cs}}$,$\models_{\mathbf{TR}}$) , *and*
3. *A is one of the actions that could be selected for execution if* f_{AS} *would be called now*

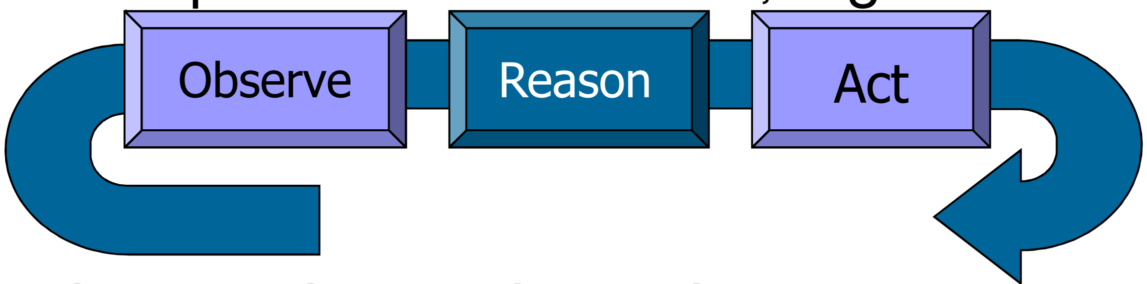
[where S = $<KB_0, F, C, \Sigma>$*]*

KGP agents' mind: an overview



Cycle theory as declarative control

- Conventional agent control:
  - Fixed sequence of transitions, e.g.



- Cycle theory determines the sequences of transitions *dynamically* and *declaratively:*
  - Flexible control, e.g. tailored to mental state/environment
- Different cycle theories give rise to different profiles of agent behaviour
- Control via cycle theories can in principle be adopted by any agent architecture

Flexible vs. Inflexible Control

- Inflexible control: E.g. Observe-think-act
- Flexible control:
  - Decide on the run what to do next, depending on
    - what you have just done, and
    - the current circumstances
  - Give different priorities to activities, e.g. give higher priority to
    - planning for “critical” goals
    - executing actions whose preconditions are known to hold
    - meeting social obligations

Examples of Flexible control

Example of transitions:

- Passive Observation Introduction (POI)
- Reactivity (RE)
- Plan Introduction (PI)
- Goal Introduction (PI)
- Action Execution (AE)
- State Revision (SR)

These allow us to represent, for example

- *observe-think-act* cycle: POI,SR, RE,PI,AE
- cycle of a *reactive agent*: POI,SR,RE,AE

Cycle Theories

- A **cycle theory, $\mathcal{T}_{cycle}$** , is a (meta-)logic program with priorities to reason about which transition(s) should be chosen next.
- The rules specify possible follow-ups of (already-executed) transitions.
- The priorities express high-level preferences of the particular agent. These characterise the operational behaviour of the agent.

Components of Cycle theories

- An ***initial* part** $\mathcal{T}_{initial}$ to reason about which transition *could* be the *first* (in some *initial state* S_0)
- A ***basic* part** $\mathcal{T}_{basic}$ to specify which transitions could be *next* (in some state S) *after a transition* that has just been executed)
- A ***behaviour* part** $\mathcal{T}_{behaviour}$ to specify which transition(s) (amongst the many possible) is (are) the preferred next one(s).
- An ***auxiliary part***
- An ***incompatibility part*** specifying incompatible transitions

Predicative representation of transitions

Within the cycle theory and
the agent behaviour, a transition

$$(T)\ \frac{S,\ X}{S'}\ now$$

is represented as an atom in the predicate T:

$$T(S, X, S', now)$$

Cycle Theories at a glance

- $\mathcal{T}_{\textit{basic}}$: cycle step rules :
  $R_{T \mid T'}(S',X')$: *T' (S',X') ← T (S,X,S', t), EC(S', X')

- $\mathcal{T}_{\textit{behaviour}}$: priority rules on cycle step rules:
  $R^{T}_{N1 \mid N2}$: $R_{T|N1}(S,X1) > R_{T|N2}(S,X2) \leftarrow BC(S,X1,X2)$

  - S, S' are states,
  - X, X', X1, X2 are inputs to transitions,
  - EC are enabling conditions that also determine the inputs to the transitions (selection operators),
  - BC are behaviour conditions,
  - > is the preference (priority) relation over rules

Agent behaviour

Cycle theories determine the **operational trace** of the agent:

$\mathbf{T_1(S_0,X_1,S_1,t_1), \ldots, T_i(S_{i-1},X_i,S_i,t_i), \ldots}$ s.t.

$\mathcal{T}_{\mathit{cycle}}, T_i(S_{i-1},X_i,S_i,t_i), now(t_{i+1}) \models_{pr} {}^{*}T_{i+1}(S_i,X_{i+1})$

where $\models_{pr}$ is entailment for LP with Priorities,
(also underlying the Goal Decision reasoning capability)

Cycle theories: example of $\mathcal{T}_{basic}$

$r_{PI|AE}(S',As): AE(S',As) \leftarrow PI(S,Gs,S',t),$
$As = f_{as}(S', t'),\ As \neq \{\},$
$\mathit{time\_now}(t')$

meaning: a Plan Introduction transition may be followed by an Action Execution transition, if there are actions to be executed (identified by the *for action selection operator* f_{as})

Cycle theories: example of $\mathcal{T}_{basic}$

$r_{POI|RE}(S',\_): RE(S',\_) \leftarrow POI(S,\_,S',\_)$

meaning: a Passive Observation Introduction transition may be followed by a Reactivity transition, unconditionally

Cycle theories: example of $\mathcal{T}_{behaviour}$

- $r_{PI|AE}(S,As) > r_{PI|N}(S,X)) \leftarrow$ not unreliable_pre(As) for all transitions $N \neq AE$

- $r_{PI|SI}(S,Ps) > r_{PI|AE}(S,As)) \leftarrow$ unreliable_pre(As)

  meaning: After Plan Introduction, the transition Action Execution should be given higher priority, unless there are actions amongst the actions selected for execution whose preconditions are "unreliable" and need checking, in which case Sensing Introduction will be given higher priority

Patterns of behaviour

- A cautious agent always checks that actions' preconditions hold before executing them
- An interruptible agent always takes into account new events in the environment (that its sensing capability can perceive)
- Focused, impatient, efficient, normal agents

Sadri ,Toni, *Profiles of behaviour for logic-based agents*, CLIMA VI, 2005

Careful and Focussed profiles: examples and motivations

- Careful profile

Plan: cancel conference registration

Observation: invalid registration

Reaction: drop Plan (as unnecessary)

Plans and Goals are re-examined "often" –

useful in dynamic and unpredictable environments

- Focussed profile

Goals: have book (10£) and have CD (15£)

Beliefs: only 20£ available

Reaction: drop one of the goals (as they cannot both be achieved)

Focus on one goal at a time –

useful if limited "resources" or "incompatible" goals

Careful profile: property

- If actions and goals never get timed-out between a SR and a RE (if next)
- Then careful agents will never react to
  - A timed-out unexecuted action
  - A timed-out unachieved goal
  - An unexecuted action whose execution is no longer needed

Focussed profile: property

Consider focussed agent f and normal agent n,
with the same state. If, for both f and n:

- the plan is empty and there are $n>1$ top-level goals
- goals $g_1 \ldots g_k$ keep on being selected till achieved
- computed plans are total (have actions only)
- no joint plan exist for $g_1 \ldots g_k$
- plans exist for each g_i in isolation

Then f will achieve at least one goal, while n will achieve none,
if, in addition:

- no actions and plans are generated by RE
- no POI is performed , and GI does not change top-level goals
- goals and actions are non-time critical

KGP model and Heterogeneity

- **Allocation of "Expertise" to agents via different modules of knowledge**
- **Personality and social behaviour via a well separated module of the agent (control)**
- **Overall Behaviour of an agent should be modularly regulated, able to exhibit different patterns of behaviour (we have seen two)**

Evaluation of KGP model

- How useful/effective is an agent model?
- What are the reasons for its design choices?
- We have identified three properties concerning the “welfare” of (KGP) agents:
  - Is the agent effective in achieving its goals? (*Goal achievement)*
  - Is the agent working towards achieving its goals? (*Progress)*
  - Is the agents effective in reacting to changes in its environments? (*Reactive awareness)*

Sadri ,Toni, *A formal analysis of KGP agents,* JELIA06

Goal achievement: preliminaries

- KGP agents go through sequences of states (generated by the application of transitions)
- A sequence $S_0, S_1, \ldots, S_n, \ldots$ is improving wrt << iff
  For each S_j there exists S_l ($l > j$) with $S_j << S_l$
- We define $<<_1$ and $<<_2$ in terms of the *number of achieved goals $A^+(S)$ in a state S:*
  $S <<_1 S'$ iff $aG^+(S) \leq aG^+(S')$
  $S <<_2 S'$ iff $aG^+(S) < aG^+(S')$
- We adopt a *subjective* notion of achievement

Goal achievement properties

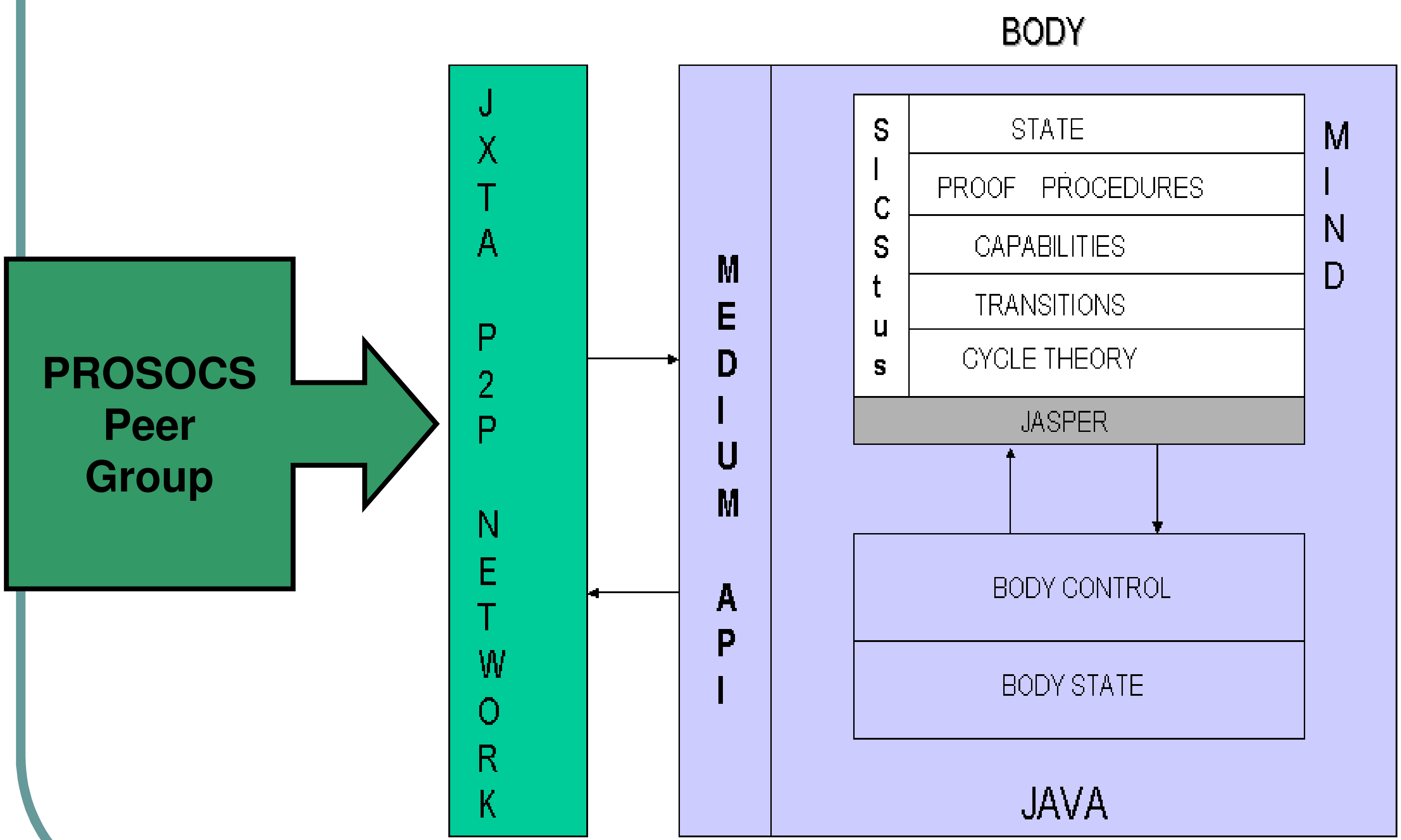
- Every KGP agent is improving wrt $<<_1$
- KGP agents may not be improving wrt $<<_2$ but definitely:
  - If S $<<_2$ S′ then there exists an intermediate (other) state S″ in which one of

    OI, PI or AE has been performed
  - Motivation of design choices (OI, PI, AE)

Properties: Some considerations

- We have studied some “welfare” properties of the KGP agent model
- Can these properties be proven of other agent models?
- Are there any other interesting properties for agents?
- Can any formal verification technique be applied to prove properties of the KGP agent model?

KGP Implementation: PROSOCS



Technologies

- SICStus Prolog for inference-based components (e.g. LP, ALP and LPP);
- JXTA for communication(e.g. Medium API for sensors/effectors)
- Java to implement the Medium API, integrate Prolog into components, and build the GUIs.

PROSOCS: Agent Mind

A number of generic components:

- normal cycle theory on selecting transitions (LPP)
- transitions calling capabilities (KGP/Prolog)
- execution of capabilities and changes on the state (KGP/Prolog)

- LPP reasoning in GORGIAS (meta-interpreter)
- LP/ALP reasoning in CIFF (meta-interpreter)
- AEC with CIFF for temporal reasoning.

Some references

- Bracciali, Demetriou, Endriss, Kakas, Lu, Stathis, *Crafting the Mind of a PROSOCS Agent*. *Applied Artificial Intelligence*, 2005.
- Mancarella, Terreni, Sadri, Toni, Endriss*, The CIFF proof procedure for abductive logic programming with constraints: Theory, implementation and experiments.* Theory and Practice of Logic Programming, 9: 691-750, 2009
- Demetriou, Kakas, *Argumentation with abduction*, 4th Panhellenic Symposium on Logic, 2003. (GORGIAS)

Part II

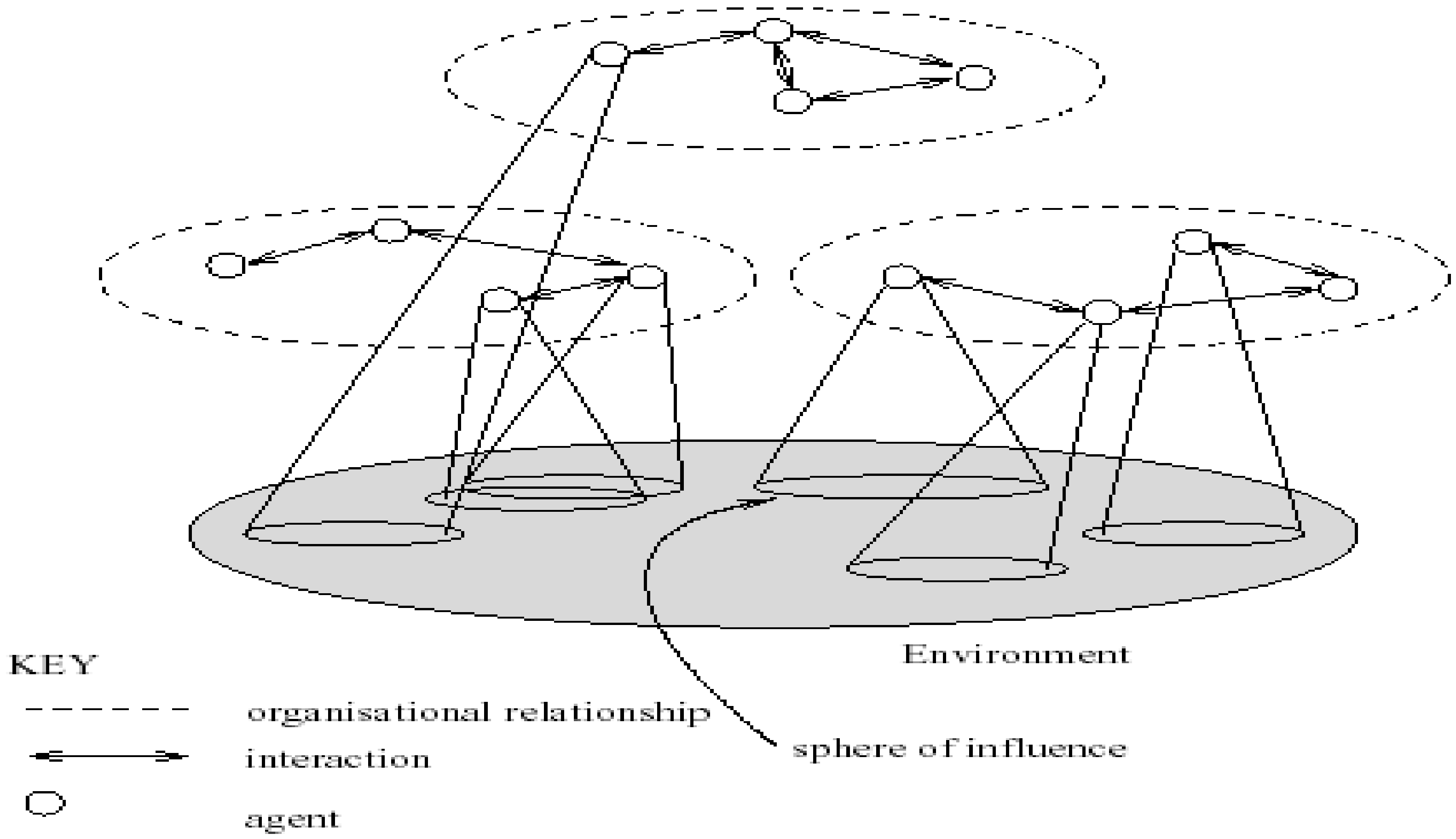
Multi-agent systems

- Communication
- Negotiation

Outline

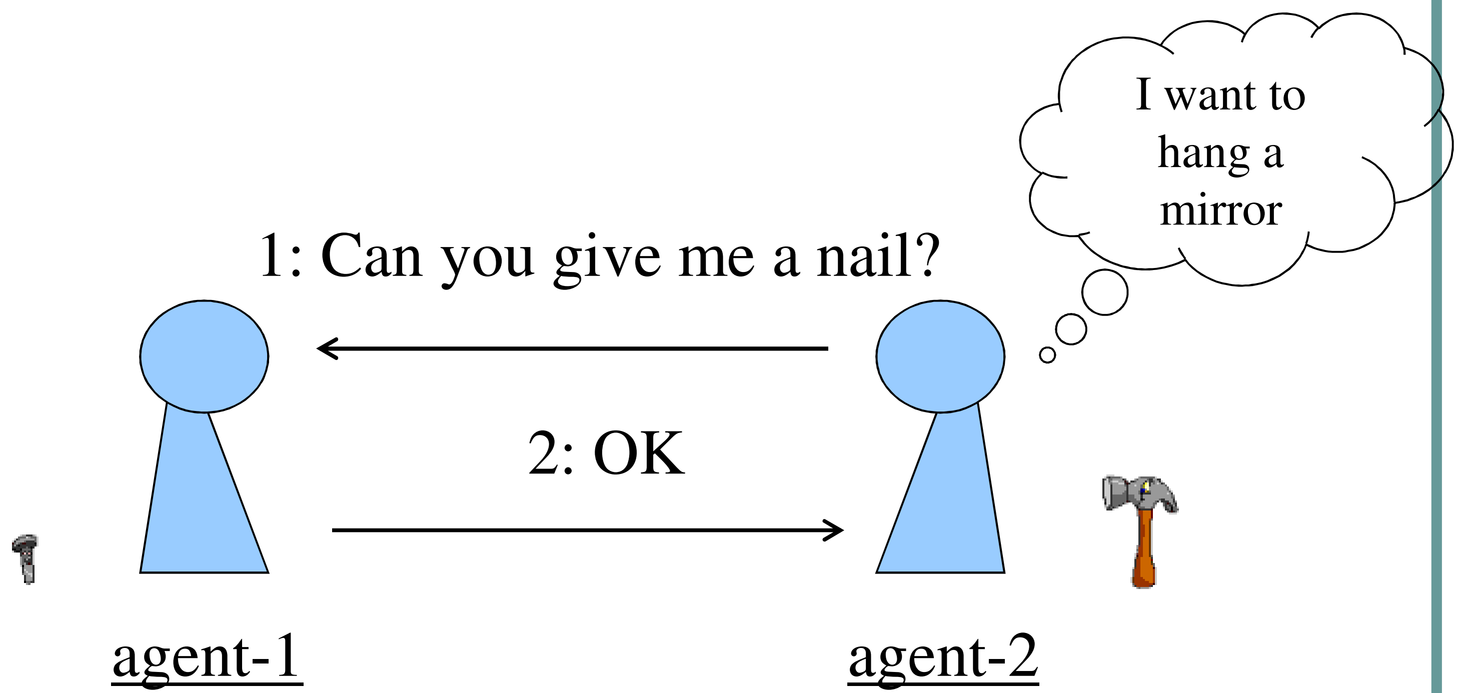
- Some background on communication
- Communication for KGP agents
- Communication for ALP agents
- Negotiation as an illustrative setting

Multiagent Systems

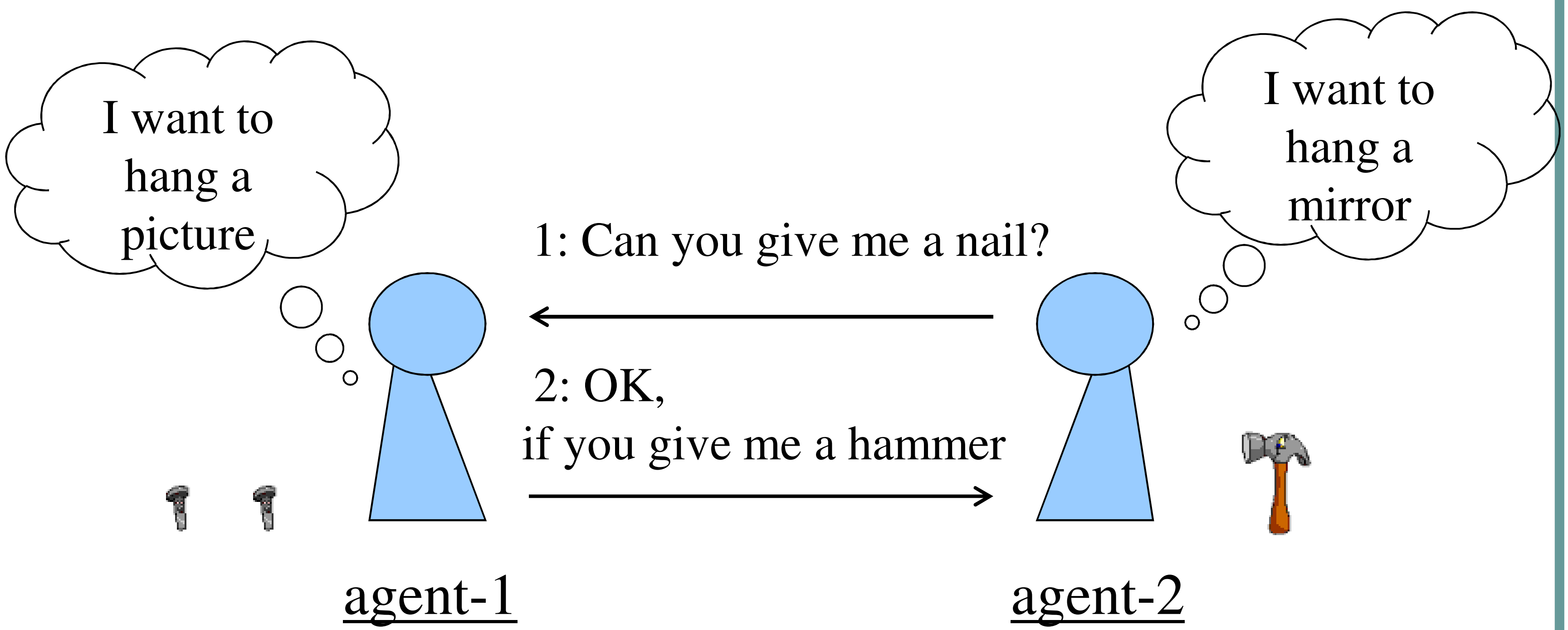


Wooldridge, *An introduction to multiagent systems*

Example (negotiation)



Example (negotiation) ctnd

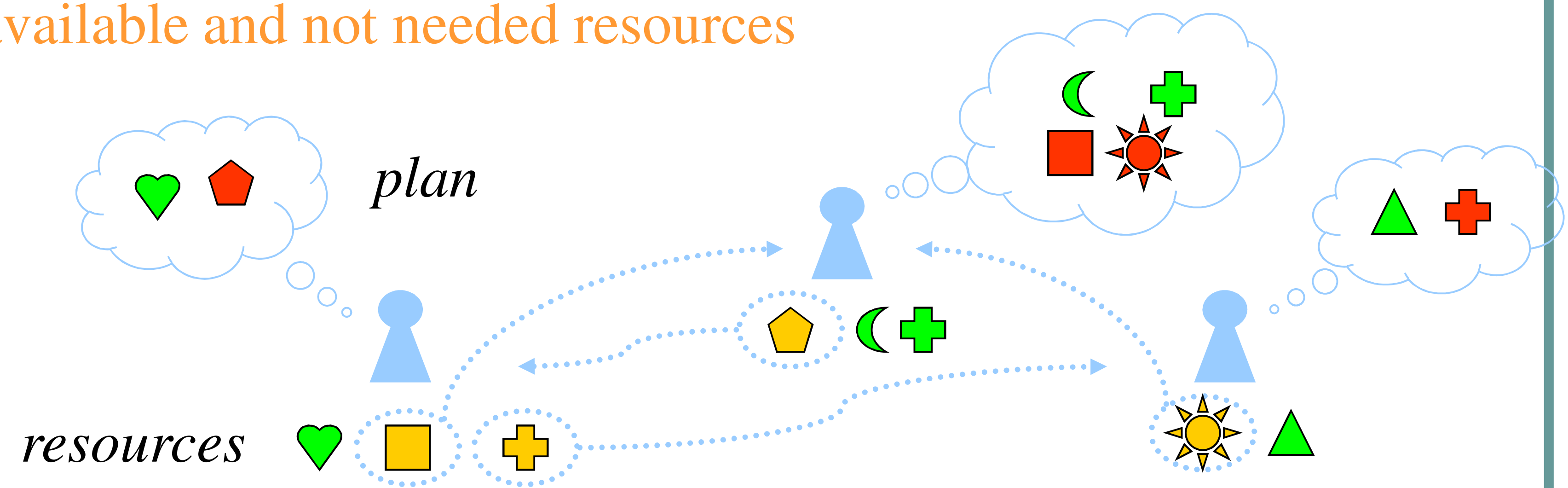


Resource Reallocation Problem

available and needed resources

missing resources

available and not needed resources



Resource sharing problem: r.r.p. over time intervals

Agent communication: Speech Act theory

- Utterance= performative+ content
  - performative = request
    content = “the door is closed”
    speech act = “please close the door”
  - performative = inform
    content = “the door is closed”
    speech act = “the door is closed!”
  - performative = inquire
    content = “the door is closed”
    speech act = “is the door closed?”

FIPA

- Foundation for Intelligent Physical Agents (FIPA) - agent standards — the centerpiece is an agent communication language (ACL)
- Basic structure follows speech act theory:
  - *performative*: 20 performative in FIPA
  - *housekeeping:*e.g., sender… but also conversation-id…
  - *content:*the actual content of the message

FIPA: Example

```
(inform
   :sender      agent1
   :receiver    agent5
   :content     (price good200 150)
   :language    sl
   :ontology    hpl-auction
)
```

Communication language: high level description

- Utterances are atoms of the form

perf(Sender, Recipients, Content, Id, Time)

  - *Sender*, *Recipients*: (sets of) agents
  - *Content* in some content language
  - *Id*: (unique) dialogue identifier
  - *Time* of the utterance (1, 2, 3, ...)
  - *perf :* performative e.g. *request, accept, …*
  - Content: *give(r), ...*
- Initial utterances
- Final (successful / unsuccessful) utterances

Inter-agent conversations/dialogues

- Flexible interaction pattern between agents
- Classification of dialogues [Walton & Krabbe, 1995]
  - Persuasion (clarification)
  - Inquiry (hypotheses proving)
  - Negotiation (settlement achieving)
  - Information Seeking (information exchange)
  - Deliberation (decision making)

  - Discovery ... [McBurney&Parsons]

Example of ACL(for negotiation)

- ***request***(*x, Y, give(R, [Ts, Te]) , Id, T)*
  used by *x* to request y a resource *R* from time Ts to time Te
- ***promise***(*x, Y, give(R , [Ts, Te],[Ts', Te']), Id, T)*
  used to propose deals: *x* will give *R to Y from time Ts to time Te* if *Y* will give *R to x from time Ts' to time Te'*
- ***refuse*- *request*** *(x, Y, give(R, [Ts, Te]), Id, T)*
- ***accept*-*request***(*x, Y, give(R, [Ts, Te]) , Id, T)*
- ***accept*-*promise***(*x, Y,*
  give(R , [Ts, Te], [Ts', Te']), Id, T)
- ***change*-*promise***(*x, Y,*
  give(R , [Ts, Te], [Ts', Te']), Id, T)

Example of dialogue

request(x, y, give(nail, [10, 11]), id, 1)

accept(y, x, give(nail, [10, 11]), id, 4)

(successful dialogue)

Another example of dialogue

request(x, y, give(nail, [10, 11]), id, 1)

promise(y, x, give(nail, [10, 11], [17, 18])), id, 2)

change-promise(x,y, give(nail,[10, 11], [17, 18])), id, 3)

refuse-request(y, x, give(nail, [10, 11])) , id, 4)

(unsuccessful dialogue)

Communication policies

- Policies as sets of logic-based rules

$(\forall)\ u[T] \wedge C \Rightarrow (\exists)\ u'[T'] \wedge TC[T,T']$

  - **u** [T] (the *trigger event*) and **u**'[T'] (the *next utterance*) are utterances *(we ignore here many details, e.g. sender)*
  - TC[T,T'] is a temporal ordering constraint between T and T', for example T < T' or T<T' < T+10
  - **C**, the *condition*, is a conjunction of literals in some logic language, equipped with an entailment operator; C is to be evaluated in a knowledge (belief) base K of the agent

Sadri, Toni, Torroni. *Dialogues for negotiation: agent varieties and dialogue sequences. ATAL01*

Resource reallocation: a simple policy

request(*X, a,* **give(R)**, Id, *T)* $\wedge$
have(R, T)
$\Rightarrow$ **accept-request**(*a, X,* **give(R))**, Id, *T1)* $\wedge$ *T1>T*

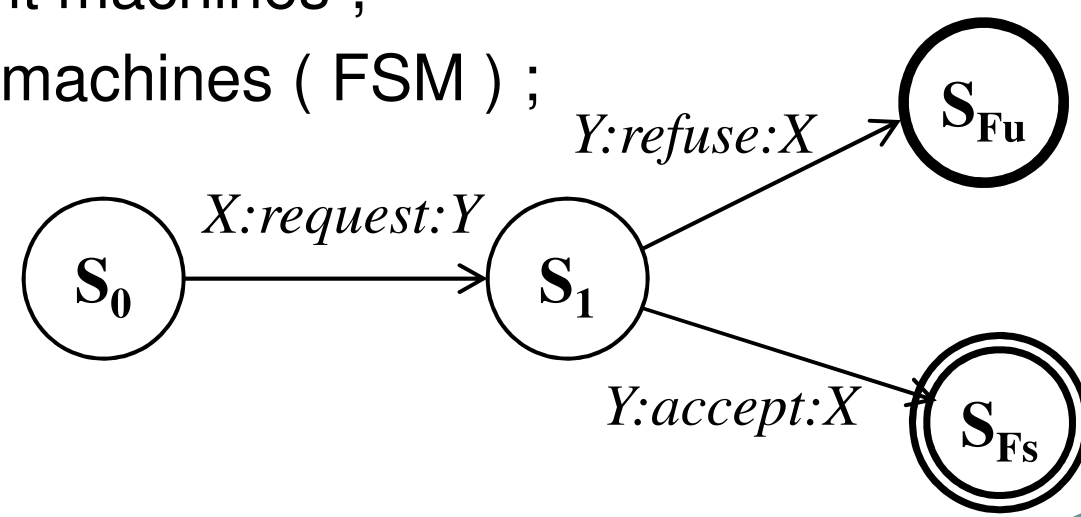
request(*X, a,* **give(R)**, Id, *T)* $\wedge$
not *have(R, T)*
$\Rightarrow$ **refuse- request**(*a, X,* **give(R))**, Id, *T1)* $\wedge$ *T1>T*

From policies to protocols

- Policies: agents' internal "rules" for (part of the) decision making about communication
  - private - encapsulated within agents' minds
  - different agents might have different policies
- Protocols: "rules" of communication amongst agents, providing a "social", non-mentalistic semantics to interaction:
  - public
  - shared amongst agents

Protocols

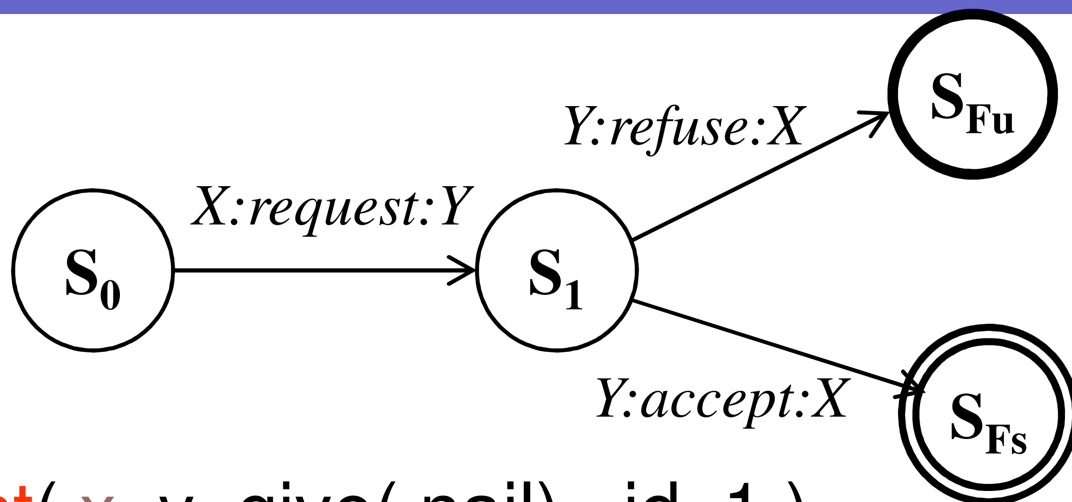
- There are several ways to define protocols:
  - input-output pairs ;
  - $\mathcal{A}$-UML diagrams ;
  - Petri Nets ;
  - commitment machines ;
  - finite state machines (FSM) ;
  - …



Conformance to protocols

- Are dialogues conformant to protocols?
- Are agents conformant to protocols? – i.e. do they utter correct utterances in dialogues?
- Are dialogues induced by policies conformant to protocols? – i.e. are agents holding those policies guaranteed to utter correct utterances in all dialogues? (a-priori conformance)

Dialogues conformant to protocols



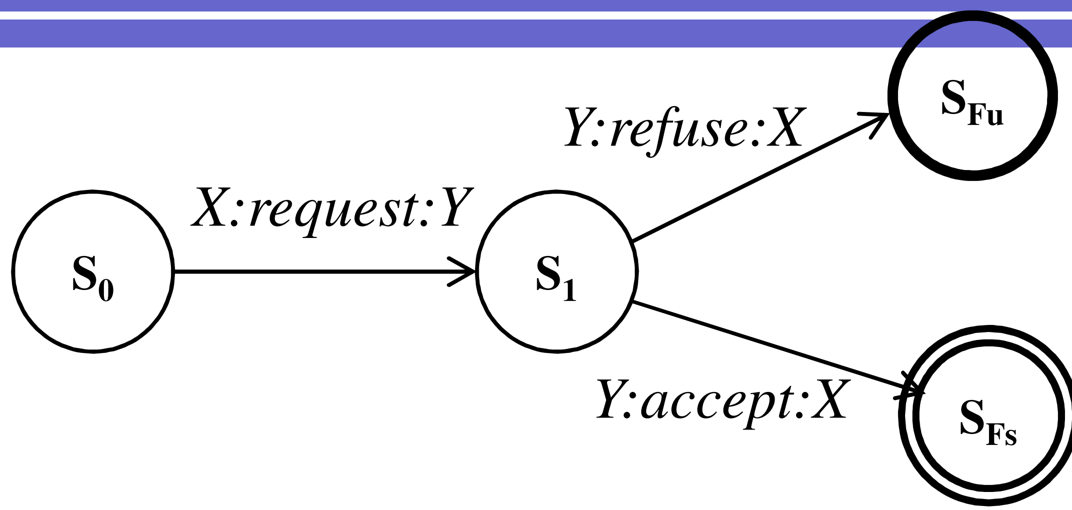
- request(x, y, give(nail) , id, 1),
  accept(y, x, give(nail)), id, 4)
  is conformant
- request(x, y, give(nail), id, 1),
  ?(y, x, give(nail)), id, 4)
  is not

[IJCAI'03b] Endriss, Maudet, Sadri, Toni,
Protocol conformance for logic-based agents

Agents conformant to protocols

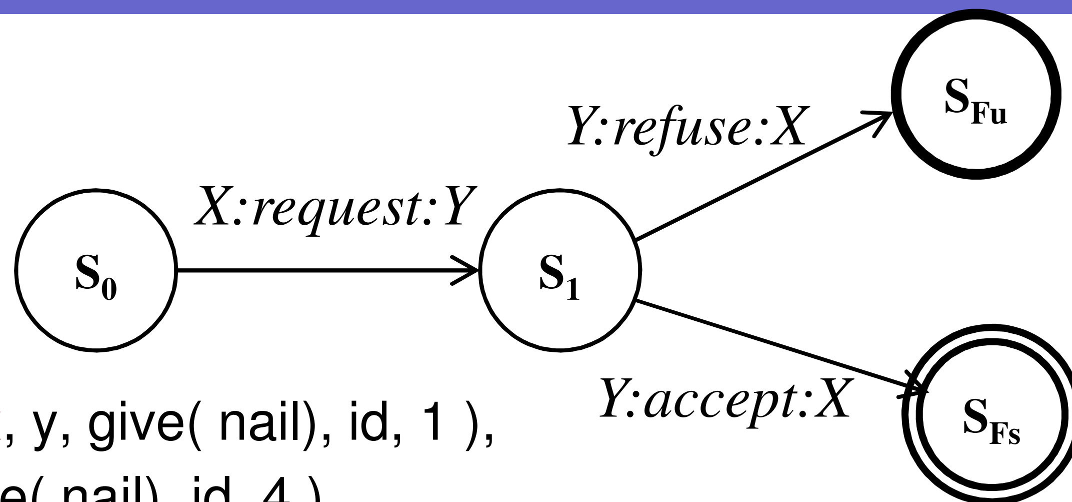
- Weakly conformant agent: *the agent never utters something "illegal" wrt the protocol – it will never give rise to non-conformant conversations while conversing with a weakly conformant agent*
- Exhaustively conformant*:* weakly conformance + *the agent always utters something correct when required by the protocol (not in final state)*
- Robustly conformant*:* exhaustively conformance *wrt to the protocol extended by a special utterance to reply to "illegal" utterances* (according to the protocol) *from other agents* (which are not conformant to the original protocol)

Agents conformant to protocols



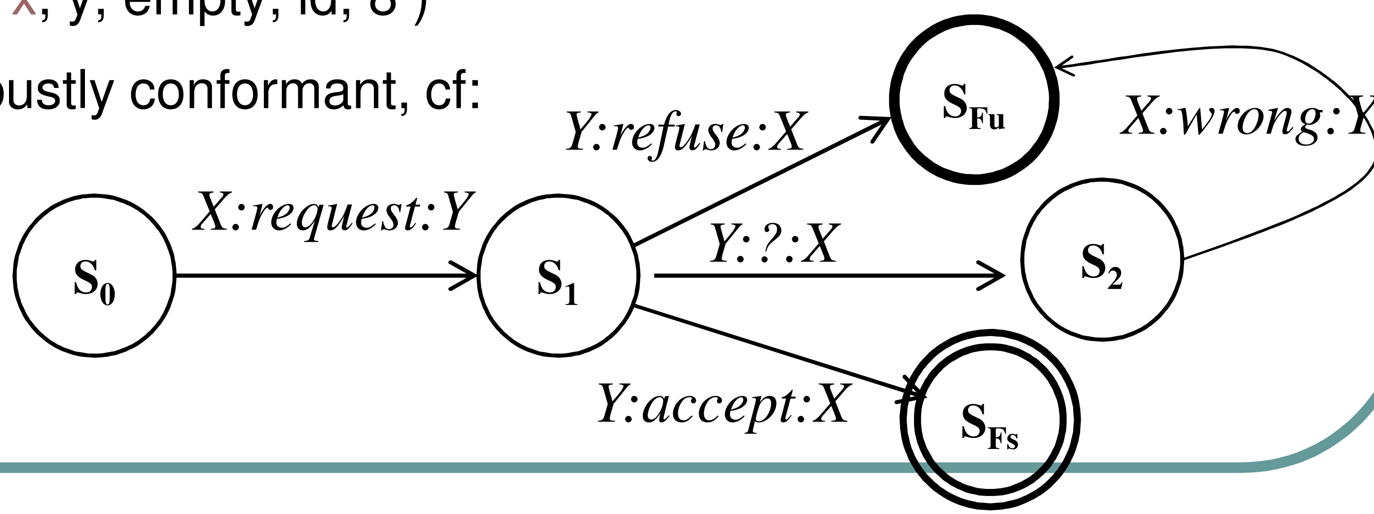
- request(x, y, give(nail) , id, 1),
  ?(y, x, give(nail)), id, 4)
  y is not weakly conformant ,
  x is not robustly conformant
- request(x, y, give(nail), id, 1)
  y is not exhaustively conformant

Agents conformant to protocols (cntd)



request(x, y, give(nail), id, 1),
?(y, x, give(nail), id, 4)
wrong(x, y, empty, id, 8)

x is robustly conformant, cf:



A priori protocol conformance

- How to design policies guaranteed to render agents (weakly/exhaustively/robustly) conformant to protocols?

A-priory conformance

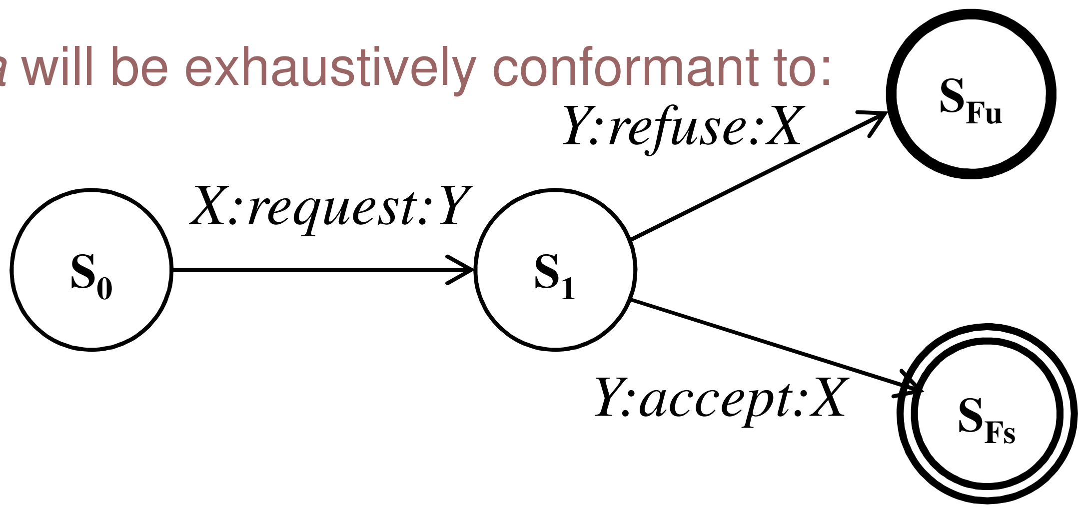
request(*X, a,* **give(R)**, Id, *T)* ∧ *have(R, T)*

⇒ **accept** *(a, X,* **give(R)**, Id, *T1)* ∧ *T1>T*

request(*X, a,* **give(R)**, Id, *T)* ∧ ***not*** *have(R, T)*

⇒ **refuse-** *(a, X,* **give(R)**, Id, *T1)* ∧ *T1>T*

Agent *a* will be exhaustively conformant to:



if its entailment notion is complete

A priori protocol conformance: result

- If logic-based representation of protocols is added to the knowledge base of (some kinds of) agents ...
- ...then these agents can be proven to be weakly conformant to those protocols (prior to their involvement in any conversations)

Communication: KGP vs ALP agents

- *KGP agents*
  - *Communicative* actions are special kinds of actions
  - *Communication policies* are reactive constraints in Kb_{react}
- Reactive constraints=integrity constraints in abductive logic programming

Abductive logic programs for communication

Abductive Logic Program =
 Logic Program (P)
 + Set of abducible predicates (A)
 + Integrity Constraints (IC)

<P,A,IC>

beliefs + past dialogues

communication language

policies (and protocols)

[JELIA'02] Sadri, Toni, Torroni.
An abductive logic programming architecture for negotiating agents.

Example

Agent a:

P_a: get(R, T) $\leftarrow$ request(a,b,give(R),d(a,b,R,T'),T')$\wedge$

accept(b,a, give(R),d(a,b,R,T'),T'') $\wedge$ T'' $\geq$ T' $\geq$ T

A_a: request (a,X, give(R),D,T), accept(X,a, give(R),D,T)

IC_a: $\varnothing$

observables

Agent b: *actions*

P_b: have(r)

A_b: accept (b,X, give(R),D,T), request (X,b, give(R),D,T),

IC_c: request(X,b, give(R),D,T) $\wedge$ have(R) $\Rightarrow$

$\exists$T'[accept(b,X, give(R),D,T') $\wedge$ T' $\geq$ T]

Example: generation of dialogue

request (a,b, give(r), _,5), accept(b,a, give(r), _,10)

a:

OBS	GOALS	ACTIONS
	get(r,0)	
	request(a,b,r,T') ∧ accept(b,a,R,T'') ∧ T'' ≥ T' ≥ 0	
		request(a,b,r,5)
accept(b,a,r,10)		
	10 ≥ 5 ≥ 0 √	

b:

ACTIONS	GOALS	OBS
		request(a,b,r,5)
	have(r) ⇒ ∃ T [accept(b,a,r,T) ∧ T ≥ 5]	
accept(b,a,r,10)		
	10 ≥ 5 √	

Advantages of abductive logic programming for communication

Dialogues generated by interleaving two abductive derivations, obtained by applying an abductive proof procedure

- Existing semantics and proof procedures (e.g. CIFF proof procedure) can be used
- Existing theoretical results (e.g. about termination) for the procedures can be exploited to prove results of communication

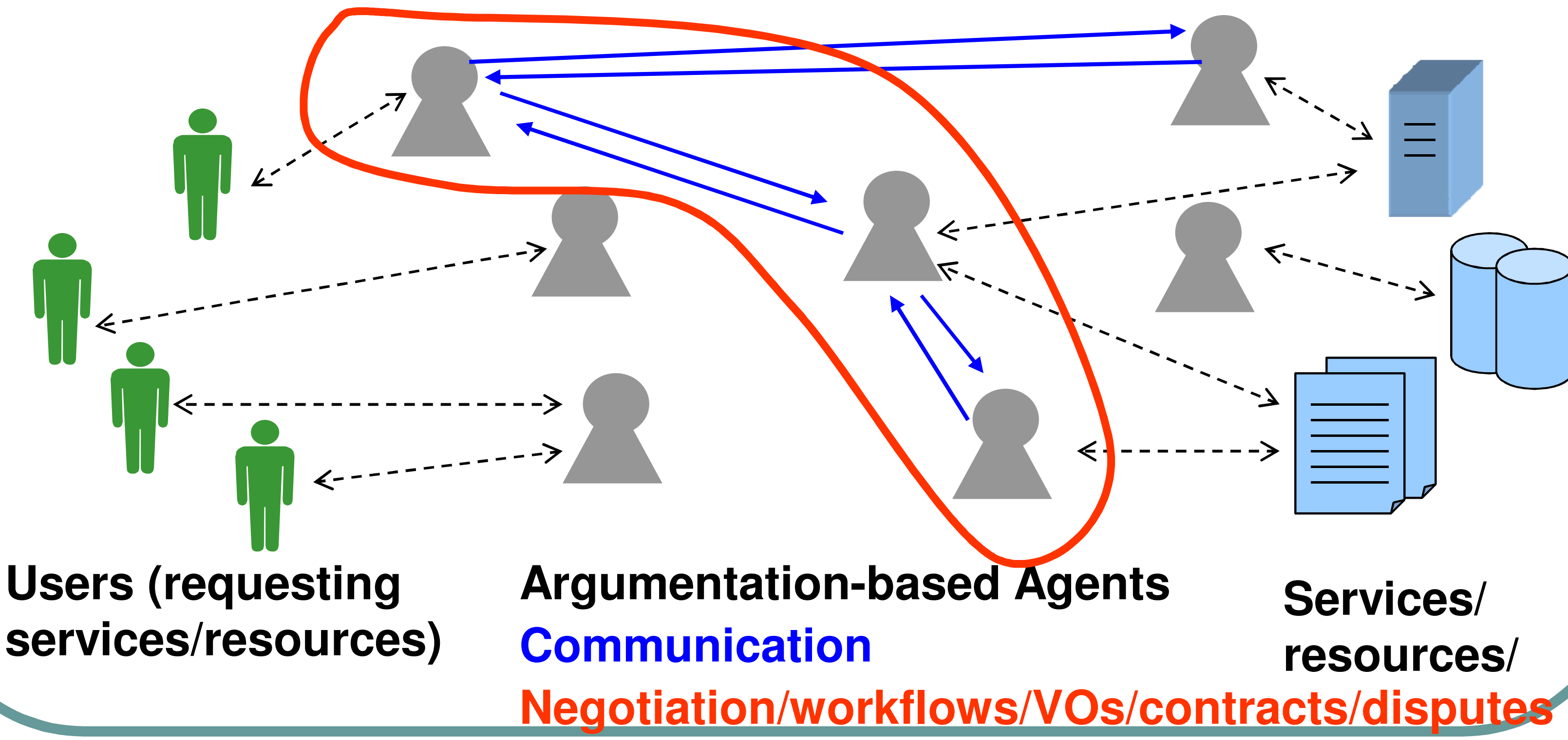
Part III

Argumentative (KGP) agents

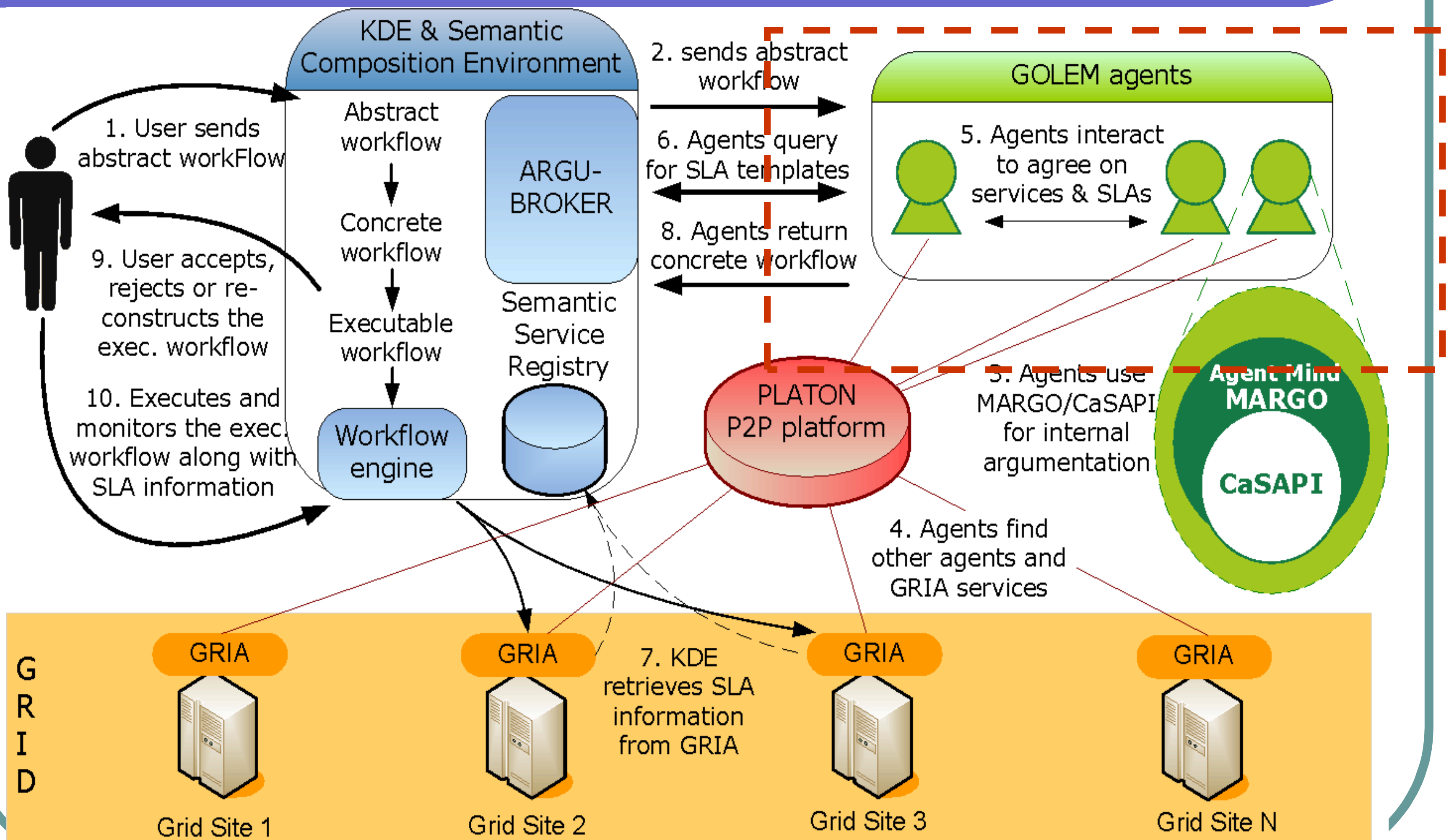
- Service-oriented architectures and Grid

Service-oriented computing

- Agent-based *semantic* grid/*service-oriented architecture*



ARGUGRID


KDE & Semantic Composition Environment
Abstract workflow
Concrete workflow
Executable workflow
Workflow engine
ARGU-BROKER
Semantic Service Registry
1. User sends abstract workFlow
9. User accepts, rejects or re-constructs the exec. workflow
10. Executes and monitors the exec. workflow along with SLA information
2. sends abstract workflow
6. Agents query for SLA templates
8. Agents return concrete workflow
GOLEM agents
5. Agents interact to agree on services & SLAs
PLATON P2P platform
3. Agents use MARGO/CaSAPI for internal argumentation
Agent Mind
MARGO
CaSAPI
4. Agents find other agents and GRIA services
7. KDE retrieves SLA information from GRIA
G R I D
GRIA
GRIA
GRIA
GRIA
Grid Site 1
Grid Site 2
Grid Site 3
Grid Site N

Scenarios

- Earth observation
  - Select appropriate sensors/satellites e.g. for dealing with oil spill
  - Combine sensors/satellites + other services (weather) e.g. for fire monitoring
- E-procurement
  - Select (combinations of) appropriate products/service to be purchased
  - Features of products/services influence business strategic benefits for the buyer

Analysis of (EO) scenarios

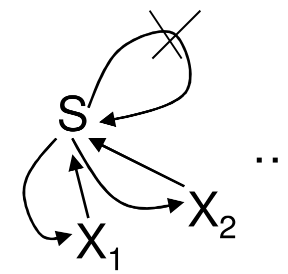
- defeasible, conflicting information/ beliefs (*it will be cloudy; it won't be cloudy; if cloudy then radar sensors*)
- mutually exclusive decisions (*sensor s1 or sensor s2*?) for the achievement of goals *(I need images every hour)*
- preferences over beliefs *(I trust weather forecast by abc more than by xyz),* over decisions (*s1 is typically more reliable than s2*), over goals *(quality of images more important then cost)*
- negotiation (*I need images every hour for a week, can I get a special price?*)

The case for argumentation

- Decision-making:
  - Alternative decisions, e.g.
    - Requestors: Which (combination of) services? From which providers? (Which protocol for asking? Which registries?)
    - Providers: Which request to accept? etc
  - Conflicting beliefs. Defeasible rules.
  - Preferences
- Negotiation
  - Justification of decisions
  - Persuasion
  - Increase the chance of success while striving for privacy

(Computational) argumentation

- Abstractly: given framework *(arguments,attack)*
  - A subset *S* of *arguments* is
    - *Admissible* iff *S* does not *attack* *S* and *S attacks* each X that *attacks* S
    - *Preferred* iff *S* is maximally admissible
    - *Grounded* iff *S* is minimal such that it contains every *a* such that *S* attacks every *X* that attacks *a*
    - *Ideal* iff *S* is admissible and contained in each preferred set
    - …



- Concretely:
  - arguments built from facts/rules
  - attack ~conflict/inconsistency/contradiction

Argumentation in philosophy and law

- Parties plead for and against conclusions
- Law:

A. Simpson is the murderer as DNA tests show
   - Simpson's blood at the murder scene and
   - the victim's blood on Simpson's glove

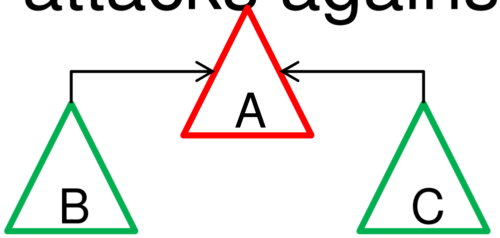
B. But the glove does not fit Simpson's hand

C. Also, the key police officer who collected the evidence is a racist

Abstract argumentation

(Dung, 1995)

- Given *(Arguments , Attacks),*
  - *Arguments:*
    - A. Simpson is the murderer as …
    - B. The glove does not fit ...
    - C. The police officer is a racist...
  - B *Attacks* A, C *Attacks* A
- Determine "winning" arguments, e.g. arguments that defeat all attacks against them
  - B and C are winning



Rule-based argumentation

- Arguments are deductions
  - of claims
  - from premises
  - using rules
  - *Simpson is a murderer*
  - *DNA shows it*
  - *If DNA shows X is M then X is M*
- Attacks may be on claims, premises or rules

Assumption-based argumentation (ABA)

- premises are assumptions
- attacks are reduced to attacks on assumptions, via a notion of contrary

Bondarenko, Dung, Lowalski, Toni 1997

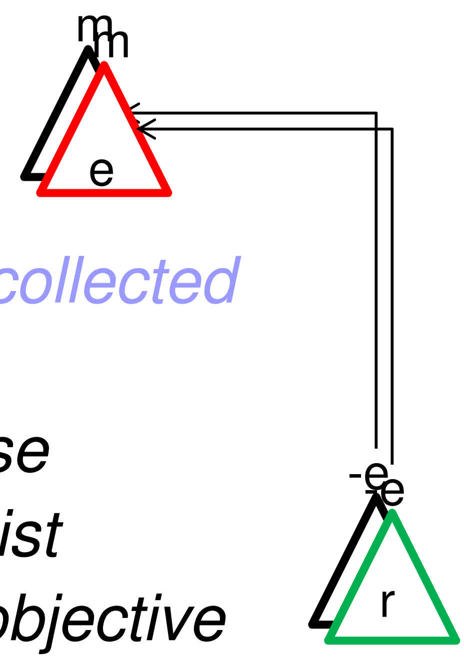
Arguments and attacks in ABA

Simpson is a murderer because

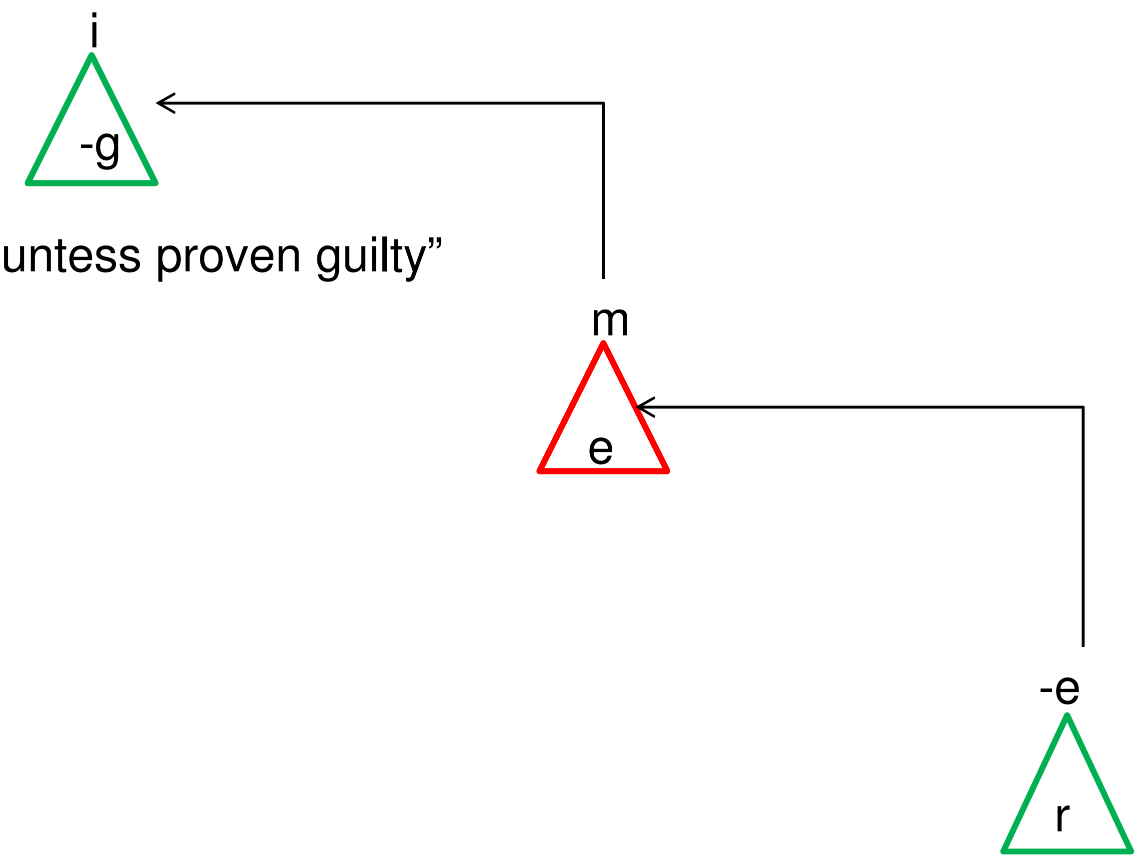
- *DNA shows it*
- *If DNA shows X is M then X is M*
- *assuming DNA from evidence correctly collected*

evidence was not correctly collected because

- *the officer collecting the evidence is a racist*
- *If X is a racist then typically X cannot be objective*
- *assuming the officer was a typical racist*



Simpson in ABA



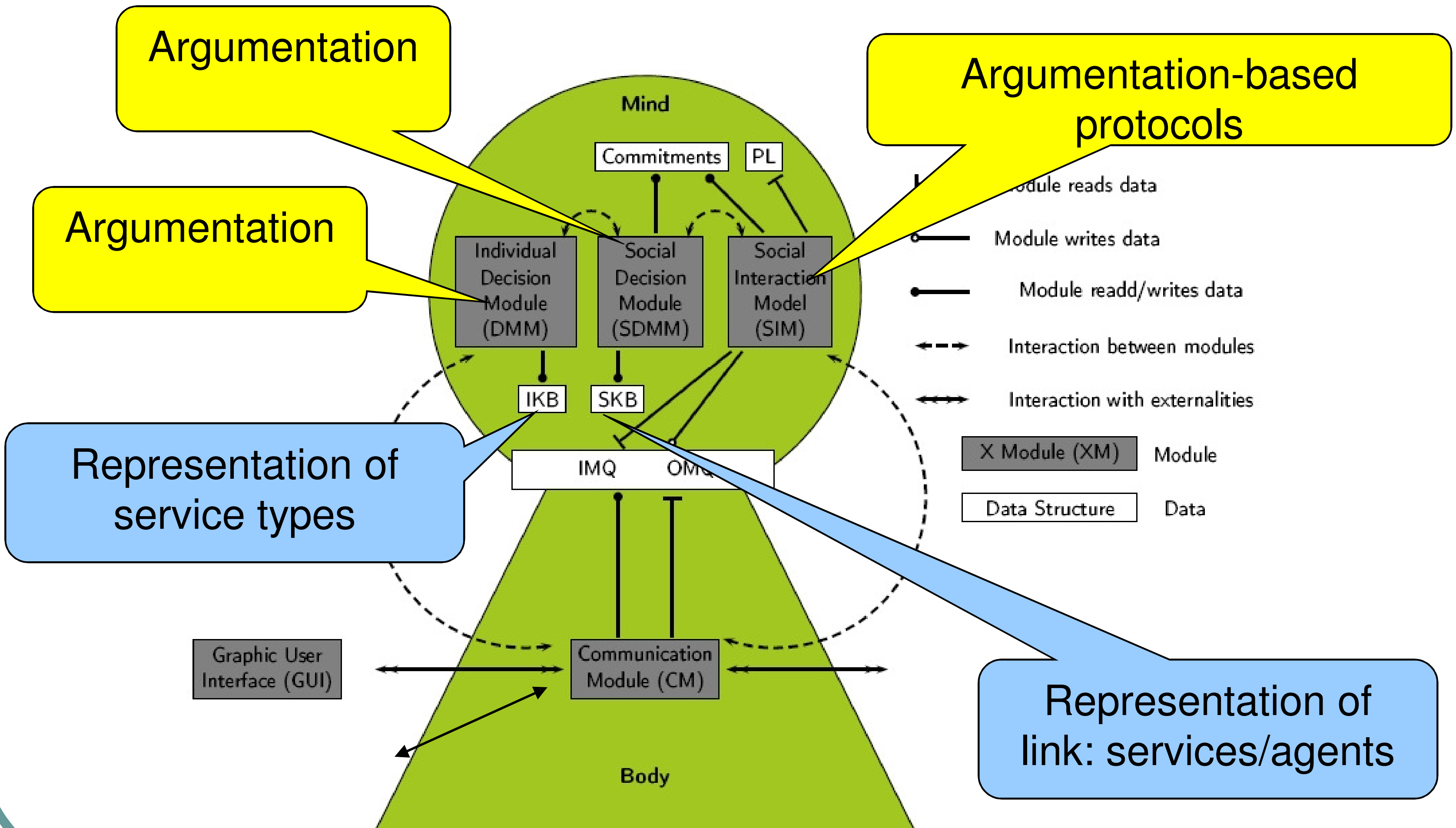
"innocent untess proven guilty"

Computation in ABA

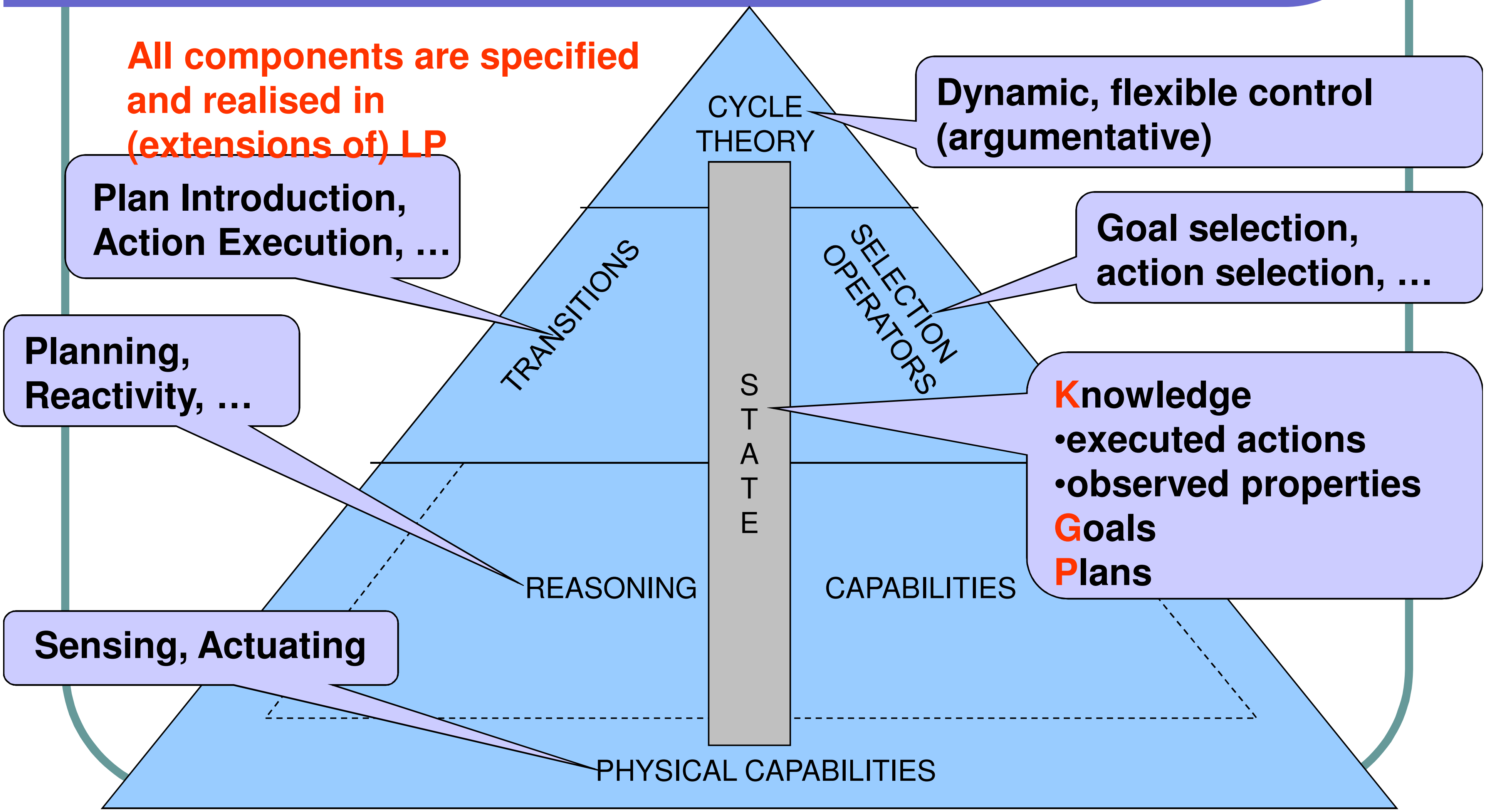
- (Various kinds of) **dispute derivations**:
  - Dispute between proponent and opponent
  - Outcomes:
    - initial claim is supported by a “winning” set of arguments or not
    - Arguments + attacks constructed during the dispute
    - Assumptions supporting the proponent’s arguments
- Prototype system: CaSAPI

Dung, Kowalski, Toni 2006, Dung, Mancarella, Toni 2007. Gaertner, Toni 2008

ARGUGRID agents (specialised KGP)

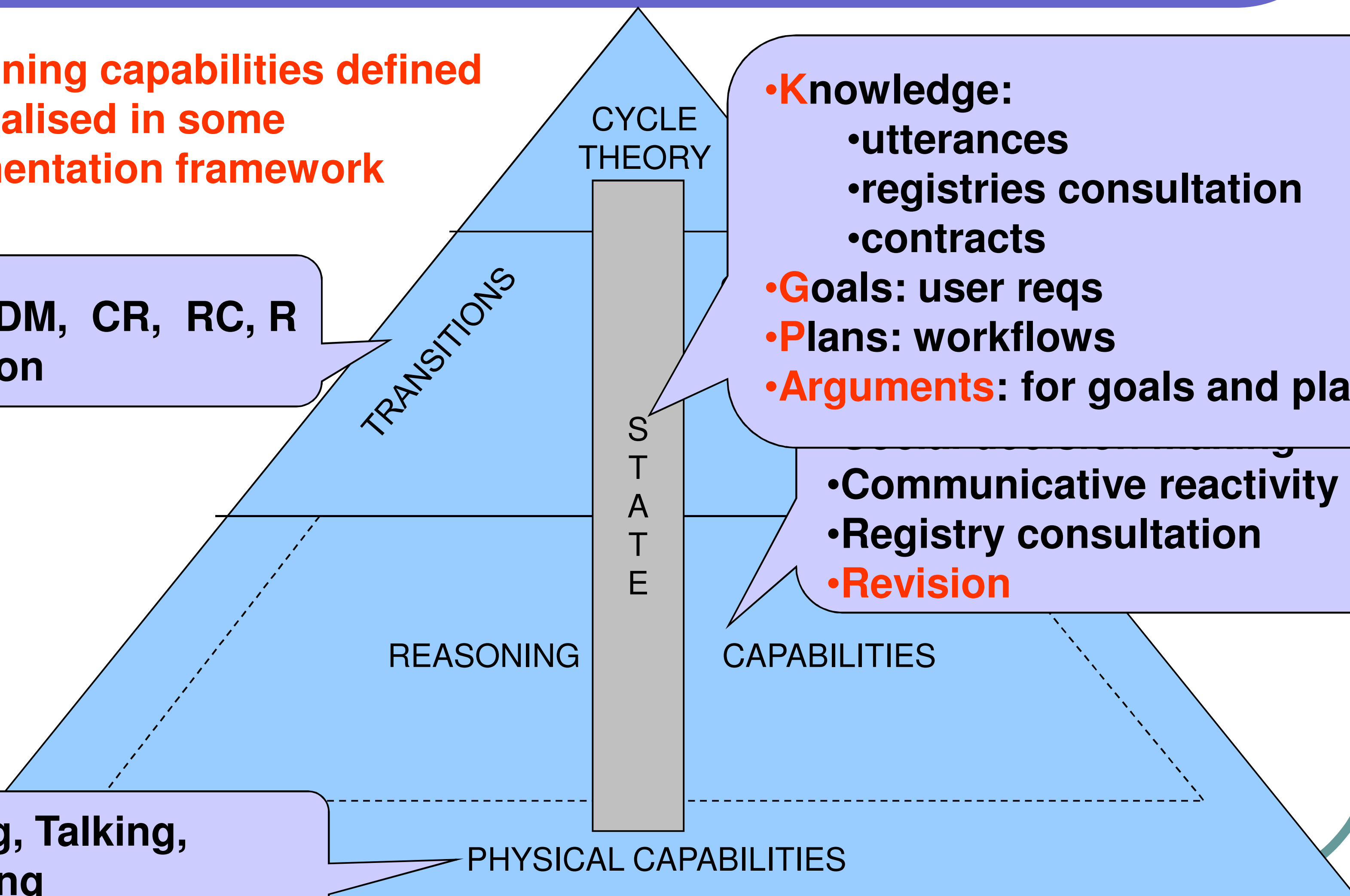


(Standard) KGP agents



Argumentative KGP agents

Reasoning capabilities defined and realised in some argumentation framework



- **ADM, SDM, CR, RC, R**
- **Li, Ta, Con**

- **Knowledge:**
  - **utterances**
  - **registries consultation**
  - **contracts**
- **Goals: user reqs**
- **Plans: workflows**
- **Arguments: for goals and plans**

- **Communicative reactivity**
- **Registry consultation**
- **Revision**

Listening, Talking, Consulting

Workflows and contracts

- *Abstract workflows* (with annotations) – outcome of *abstract decision making* reasoning capability
  - satellite(S1, I1) & processing-software(S2, I1,I2) & jpeg-format(I2)
  - computer-system(S1) & internet-provider(S2)
- *Concrete workflows* (with annotations) – outcome of social decision making+registry consultation reasoning capabilities
  - satellite(meteosat, I1) & processing-software(such-and-such,I1,I2)
  - computer-system(abc) & internet-provider(wind)
- *Contracts*: workflows + “contractual features” (e.g. cost, delivery date) – outcome of communicative reactivity reasoning capability

Registries

- Registry query language, e.g.
  - *consult(agent-such-and-such,*
    registry-such-and-such,
    Query)
  - Query may be "is there a satellite providing jpeg images"?
- Determine the registry consultation reasoning capability

Decision-making for e-procurement

- ABA:
  - features of services to purchase
  - uncertain/customisable features in services on offer
  - links from features to benefits for the buyer
  - "control information"
- e.g. *rules* may include (s_5,s_8 concrete services)
  - $f_1(s_5)$ $f_2(s_8)$
  - $f_2(S) \leftarrow$ guarantee(S)
  - $b(S) \leftarrow f_1(S), f_2(S)$,choose(S)
  - not-choose(s_5) $\leftarrow$b(s_8),not-b(s_5)

    not-choose(s_5) $\leftarrow$choose(s_8)

  with *contrary of:* choose(s_5)=not choose(s_5)

  not-b(s_5)= b(s_5), …

assumptions

Decision-making for e-procurement

- ABA framework

  $f_1(s_5)$ $f_2(s_8)$ $f_2(S) \leftarrow$ guarantee(S)

  $b(S) \leftarrow f_1(S), f_2(S)$, choose(S) ⟵ contrary: not choose(S)

  not-choose(s_5) $\leftarrow$ choose(s_8)

  not-choose(s_8) $\leftarrow$ b(s_5), not-b(s_8) ⟵ contrary: b(s_8)

- arguments

  1: {choose(s_5), guarantee(s_5)} $\vdash$ b(s_5)

  2: {choose(s_8)} $\vdash$ not-choose(s_5)

  3: {choose(s_5),guarantee(s_5),not-b(s_8)} $\vdash$ not-choose(s_8)

- attacks: *2 attacks 1, 3 attacks 2*
- admissible arguments =
  optimal choice+contracts (customisable features)

Decision-making for e-procurement

- arguments
  1. {choose(s_5), guarantee(s_5)} ├ benefit(s_5)
     "choosing some concrete offer (of a service) will provide some given benefit if that offer is extended with some additional (contractual) feature" …..argument in favour of a specific offer (s_5)
  2. {choose(s_8)} ├ not-choose(s_5)
     "choosing some offer (of a service) is a reason against choosing some other offer" …..argument against of a specific offer (s_5)
  3. {choose(s_5),guarantee(s_5),not-benefit(s_8)} ├not-choose(s_8)
     "choosing some (suitably extended) concrete offer (of a service) giving some benefit is a reason against choosing some other offer without that benefit"
- attacks: *2 attacks 1, 3 attacks 2*
- "admissible" arguments =
  optimal choice+contracts (customisable features)

Contract negotiation

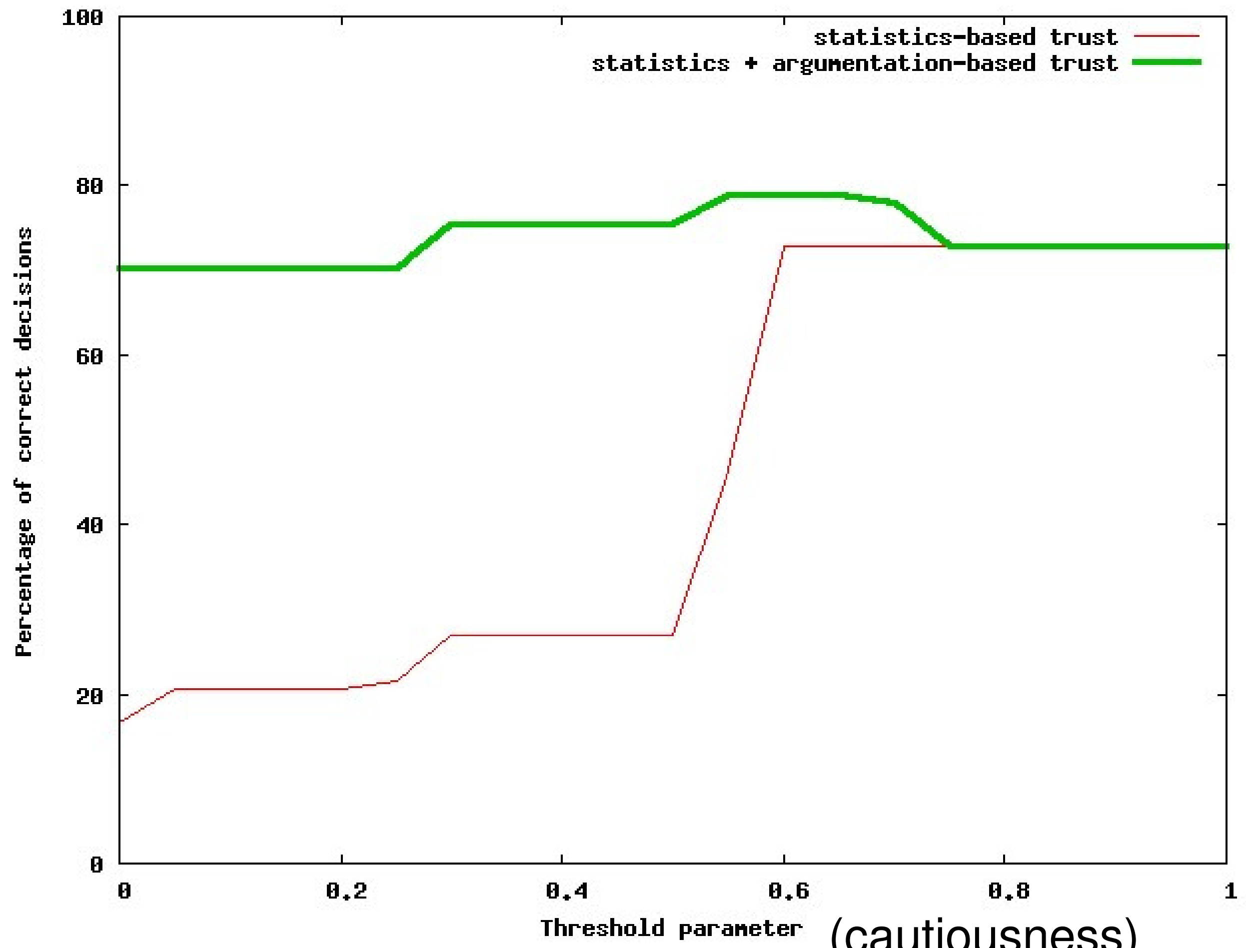
- Two agents, a buyer and a seller, each using
  - an argumentation framework describing
    - how to achieve “structural” goals (e.g. which satellite) and “contractual” goals (e.g. cost)
    - Uncertainties
    - Defeasible rules
  - Ranking of goals (preferences)
- Two-phase negotiation:
  1. (Sceptical preferred) argumentation semantics (equivalent to minmax preference for structural goals) for deciding services
  2. Negotiation protocol (of alternating offers and counter-offers) leading to agreement (using a Nash equilibrium strategy)

Argumentation for trust

- trust = willingness of an entity (evaluator) to engage in a "risky" relationship with another entity (target)
- Hybrid approach: belief function combining
  - Statistics on past behaviour of target
  - arguments (according to their strength) about predictable trustworthiness of target
    - one argument for trusting if contract by target to evaluator has clause guaranteeing QoS
    - otherwise one argument for not trusting
    - optionally one argument against trusting if, in the past, contract clause was most often violated

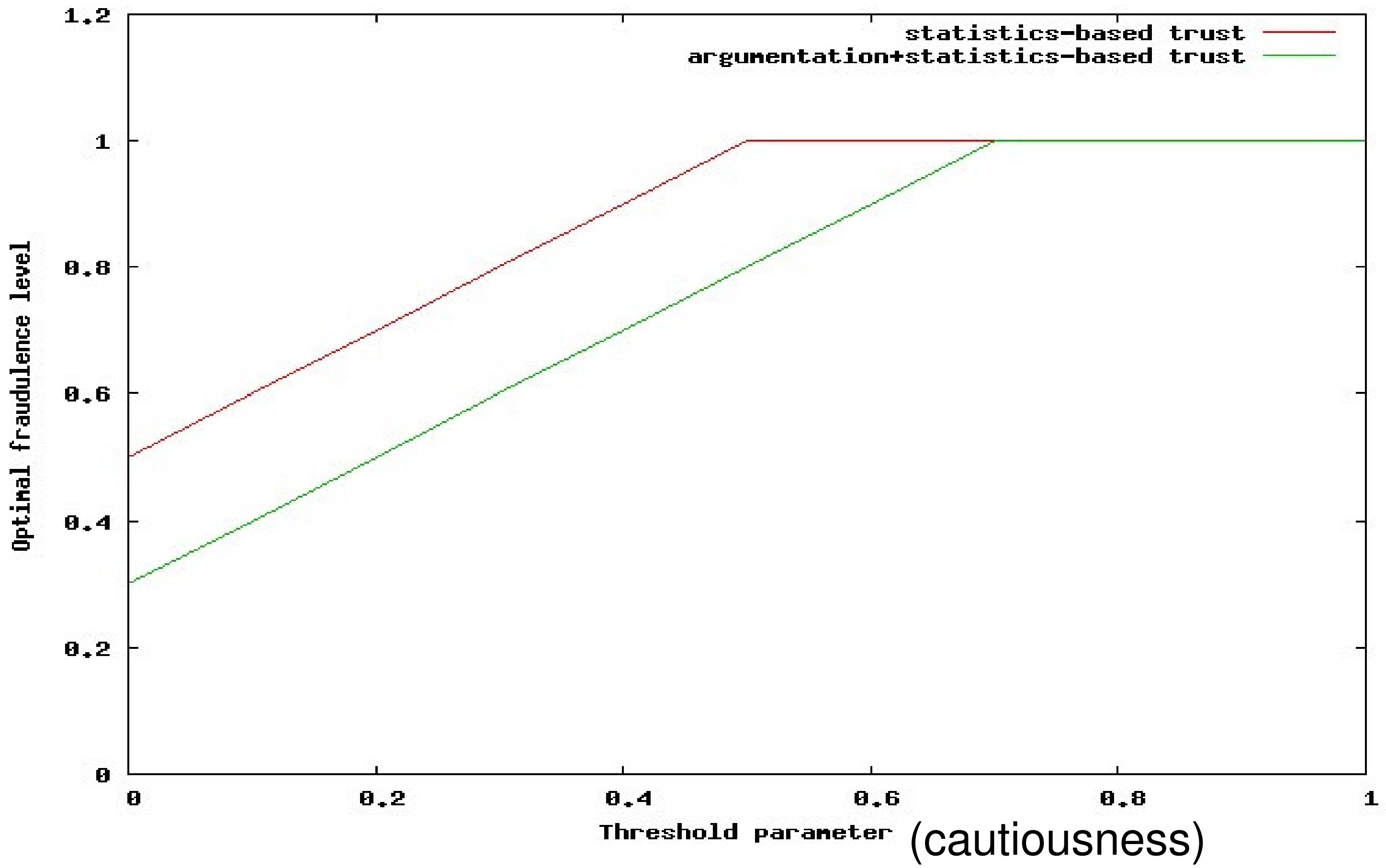
Matt, Morge, Toni, AAMAS10

Higher percentage of correct decisions (by evaluator) using argumentation



(cautiousness)

Non-fulfilment of contractual agreements (by target) is reduced by argumentation



Summary

I. Agents
   - Logic- and LP-based approaches
   - KGP agents

II. Multi-agent systems
   - Communication
   - Negotiation

III. Argumentative (KGP) agents
   - Service-oriented architectures and Grid

Acknowledgments

Thanks to

- All my co-authors
- the European Commission:

http://www.argugrid.eu

- The Royal Academy of Engineering (UK)