PROSOCS: A platform for building software agents in computational logic

---

Kostas Stathis

CITY/UNIPI

joint work with UCY, IC, UNIPI

Vienna, 14 April 2004

Overview

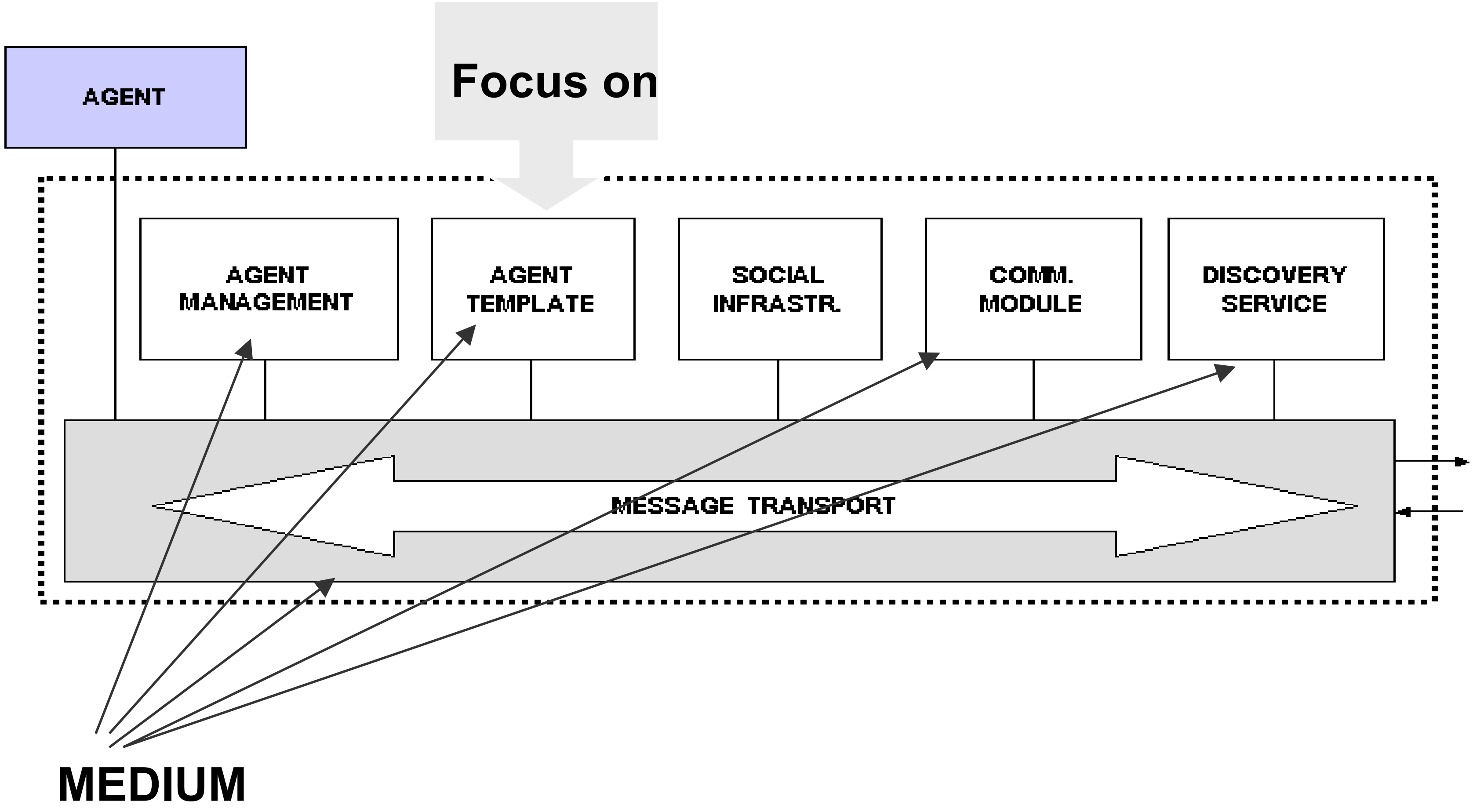
- Motivation
- Reference Model
- Agent Architecture
- Implementation
- Demo
- Conclusions

Motivation

SOCS Project (under Global Computing (GC) initiative of 5th Framework):

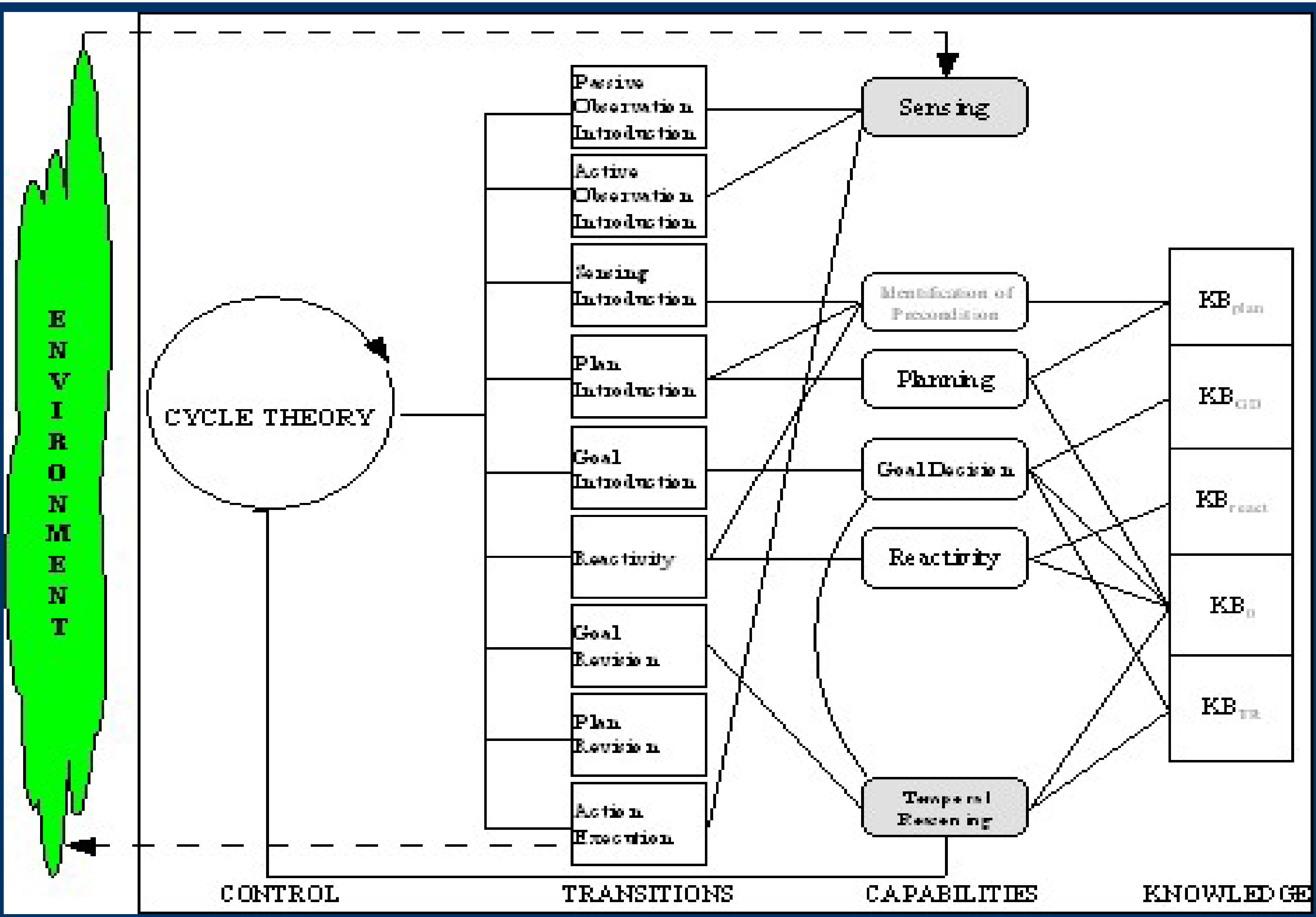
- develop formal/logical models of societies of agents in Global Computing environments;
- investigate the use of Computational Logic (as in Logic Programming) to produce executable specifications and verify properties of the resulting systems;
- integrate existing CL work and extend it;
- build a prototypes according to the models and perform experiments.

Reference Model

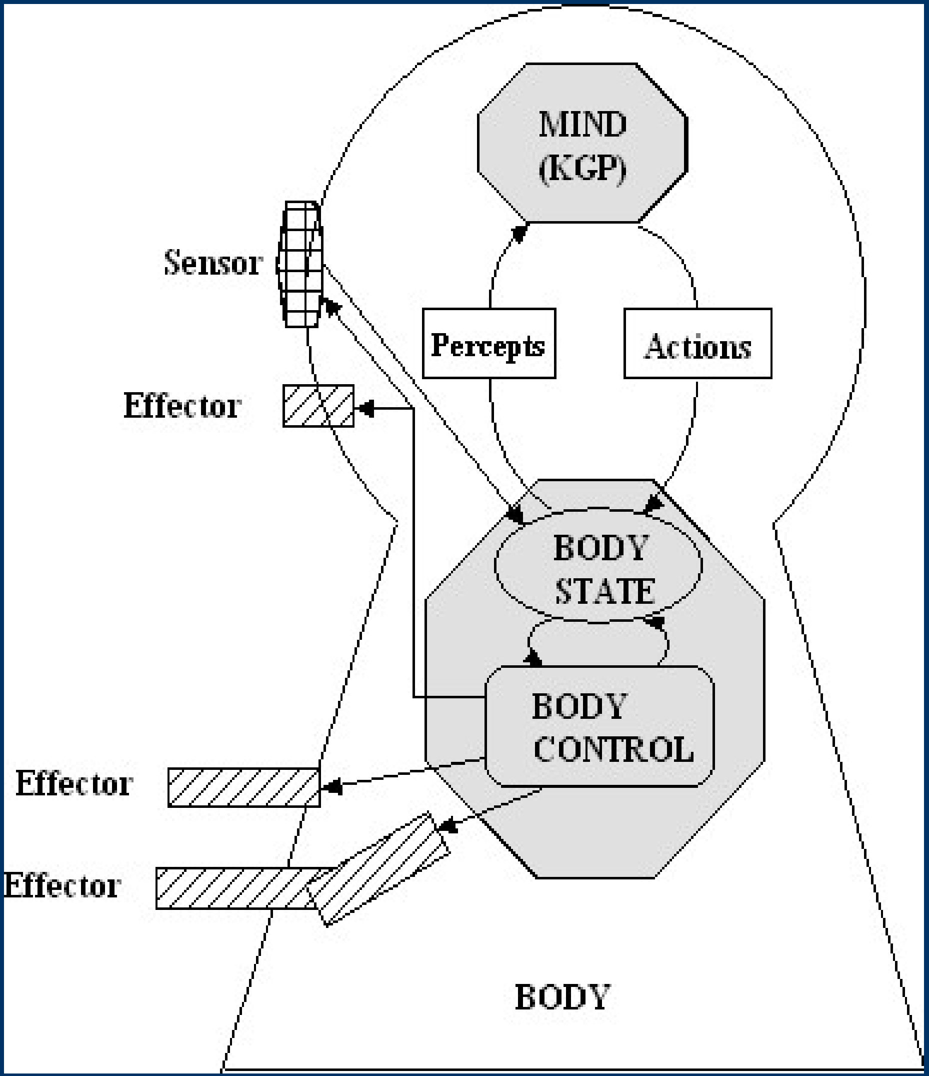


The KGP Model

?

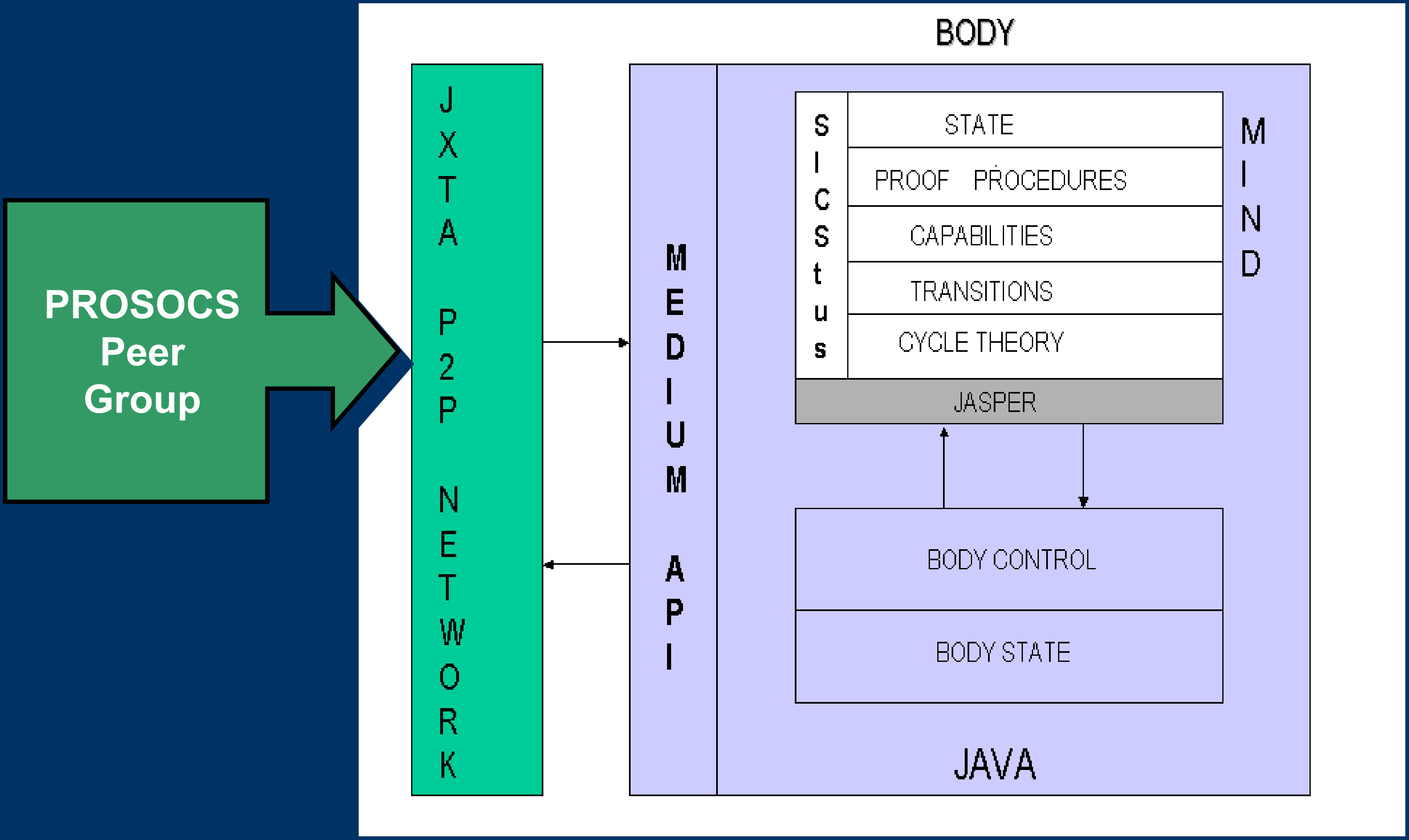


Agent Template



- work builds upon *Steiner et al* (1991), *Haugeneder et al* (1994), *Bell* (1996), *Huang et al* (2001);
- body and mind function as co-routines, i.e. concurrent thinking and action;
- interruptability.

Implementation: Architecture



Implementation: Body Control

```
private void bodyControl() {
    do {
        BodyAction nextAction = askActionFromMind();
        if (nextAction != null)
        switch (isOfActionType(nextAction)){
                case SENSING: doSee(nextAction); break;
                case COMMUNICATIVE: doSpeak(nextAction);break;
                case PHYSICAL: doEffectors(nextAction);break;
        }
        Percepts nextPercepts = sensors.passiveObservation();
        if (nextPercepts != null) tellMind(nextPercepts);
    } while (!stopped);
}
```

Implementation: Mind

In general, the mind uses a number of generic components:

- 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);
- ALP reasoning in CIFF (meta-interpreter);
- AEC for temporal reasoning.

Implementation: Mind (cntd)

To specify a cycle theory (CT) we typically want to say:

- *After perceiving a communication act (using passive observation (PO) in KGP) the agent should prefer to introduce a goal to react (using Goal Introduction (GI) in KGP).*

PROSOCS uses Gorgias (LPP) to interpret CT rules:

```
ct_rule(prefer(step(‘GI’,_), step(_,_)), prefer(step(‘GI’,_), step(_,_)), [] ):-
        last_transition(‘PO’, Obs),
        comm_msg(Obs).
```

Demo

Shows exchange of info in two simple settings:

- *Setting 1*: communicative behaviour of agents respects the social protocols;
- *Setting 2*: communicative behaviour of agents violates the social protocols.

Demo: Context

(1) A person arrives at a train station (San Vincenzo scenario):

- his agent « f » asks the train station’s agent « svs » for the information about the arrival of a train to Rome; « svs »’s reply conforms to the social rules (FIPA query_ref protocol);
- the society « s0 », which is situated in the train station, checks that interaction is conformant.

(2) Same as 1, but:

- when « f » asks for train info, « svs »’s reply violates the social rules.
- the society « s0 » shows that interactions contain violations.

Programming an agent (1) (2)

KBreact:

```
[observed(C, tell(C, svs, query_ref(Q), D, T0), T1)] implies
[assume_happens_after(tell(svs, C, refuse(Q), D), T2, T1)].

[observed(C, tell(C,svs,query_ref(Q),D,T0),T1), holds(have_info(Q,I),T1)]
implies [assume_happens_after(tell(svs, C, inform(Q,I),D),T2,T1)].
```

KBplan:

```
precondition( tell(svs,C, inform(Q,I), D),  have_info(Q,I) ).
holds_initially( have_info(arrival_time(tr123),  10.32) ).
executable( tell(svs,C,Subject,D) ) :- not (C=svs).
```

Demo …

Conclusions

PROSOCS allows the building of agents whose:

- mind is developed using ALP and LPP with extensions integrated according to the formal model KGP;
- body allows the agent to interact in a networked environment implemented on top of the P2P JXTA project;
- interactions can be checked for conformance by a society infrastructure (talk by Chesani this afternoon);
- future work involves experimenting with real applications and proving properties.