MADMASS

Massively Distributed Multi Agent System Simulator

Vittorio A. Ziparo

ziparo@algorithmica.it

Outlook

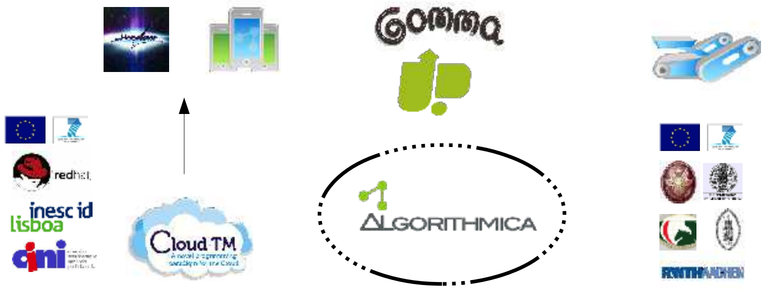
- Introduction
- Agents in the Cloud
- Behavior Modeling

Introduction

Algorithmica

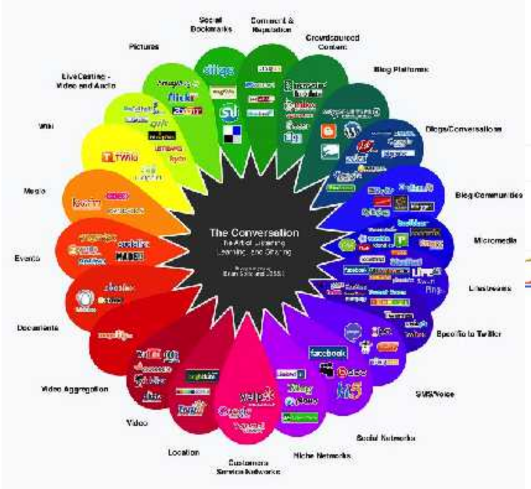
- research-oriented start-up
- founded by me, Cottefoglie and Farinelli*
- team from Lab RoCoCo (DIS, Sapienza)
- 3 research engineers + 2 PhDs*

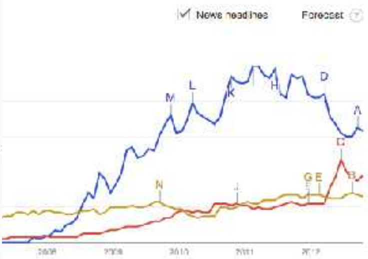
What we do



Google's hybrid approach to research
A. Spector, P. Norvig, and S. Petrov
Communications of the ACM, 2012, vol. 55, no. 7

Where is software going?





Google chrome
Operating System

Software as a Service

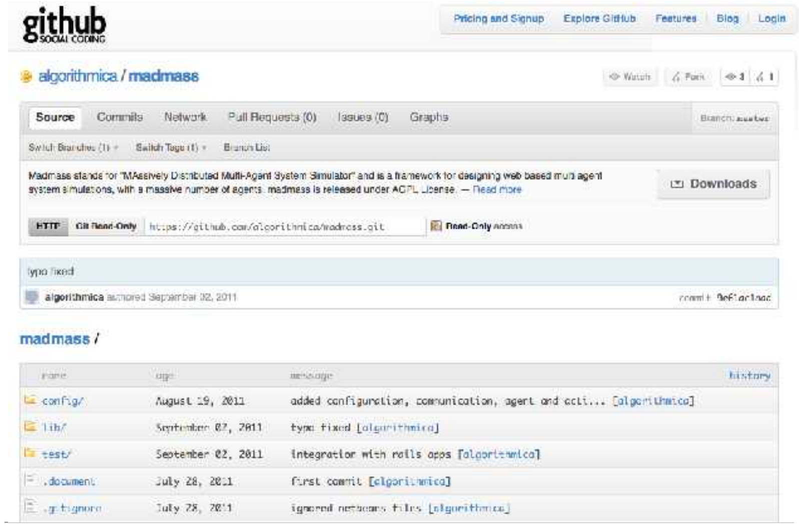
1. clients: browser and mobile apps
2. servers: distributed system running on IaaS/PaaS
3. dev: established tools , e.g., Ruby on Rails
   - Berkley SaaS course on coursera.org

... but apps are not just documents! They can become though to develop because of design complexity and scalability requirements

MAS for SaaS

- Agent-Oriented Software Engineering (AOSE)
  - complex distributed systems [Jennings, 99]
  - has a rich state of the art [Wooldridge,01]
- AOSE can simply development of SaaS
- MAS can greatly benefit from Cloud Computing

MADMASS Project



github SOCIAL CODING

Pricing and Signup | Explore GitHub | Features | Blog | Login

algorithmica / madmass

Source | Commits | Network | Pull Requests (0) | Issues (0) | Graphs

Switch Branches (1) | Switch Tags (1) | Branch List

Madmass stands for "MAssively Distributed Multi-Agent System Simulator" and is a framework for designing web based multi agent system simulations, with a massive number of agents. madmass is released under AGPL License. — Read more

Downloads

HTTP | Git Read-Only | https://github.com/algorithmica/madmass.git | Read-Only access

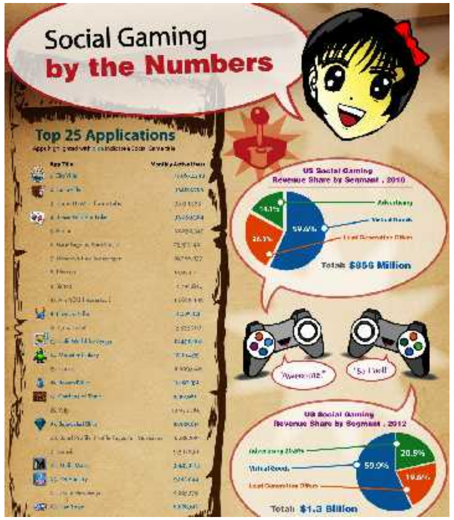
typo fixed

algorithmica authored September 02, 2011

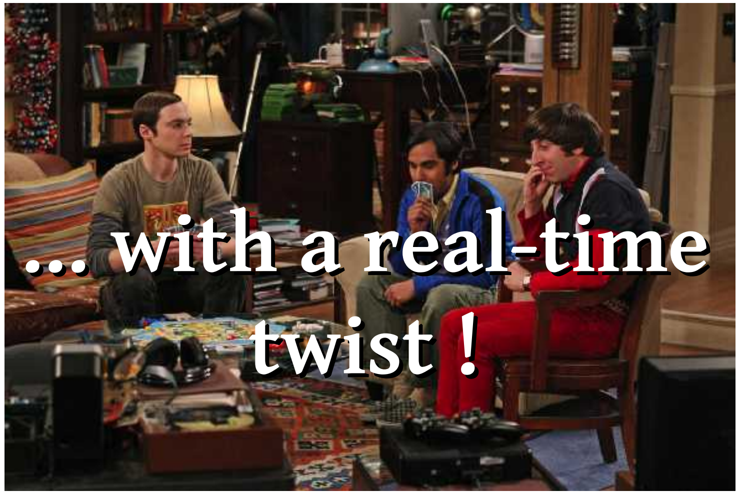
madmass /

name	age	message	history
config/	August 19, 2011	added configuration, comunication, agent and acti... [algorithmica]	
lib/	September 02, 2011	typo fixed [algorithmica]	
test/	September 02, 2011	integration with rails apps [algorithmica]	
.document	July 28, 2011	first commit [algorithmica]	
.gitignore	July 28, 2011	ignored netbeans files [algorithmica]	

Use Case: Social Gaming



Board Games



In action: the Harvestar

Agents in the Cloud

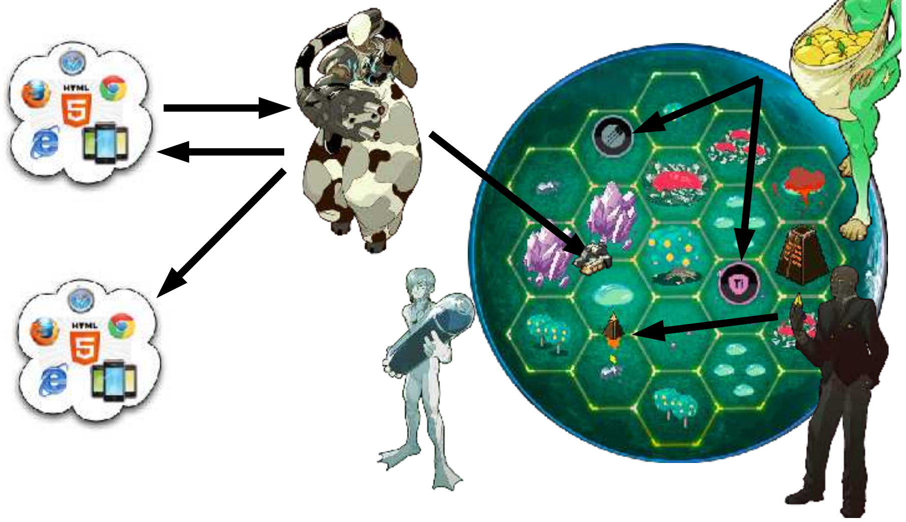
AO Programming

– Environment

– Agents (if necessary)

– User Interfaces

Example



User Interfaces

- js API + HTML5
- agent communication
  - ask_agent
  - on_percept
- GUIs not web pages (coming soon)
  - 1 page apps
  - HTML templating (Web Sockets)

Environment

– Domain Data Model (AR, Fenix-DML)

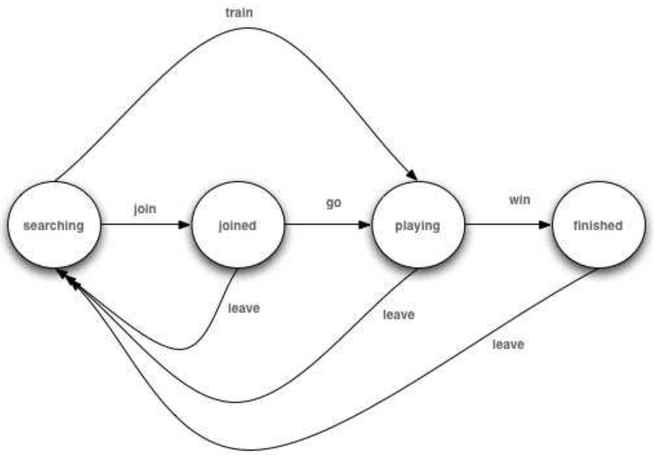
– Actions

- Preconditions
- Effects
- Percepts

Environmental Agents

- proxy agents
- periodic agents
- event-driven agents

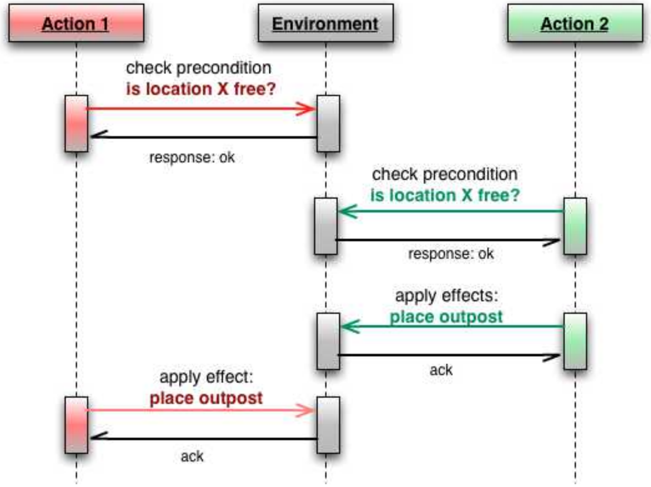
FSA constraints



Execution Correctness

"[...] if the assumptions (pre-condition) underlying the procedure become false while the procedure is executing, then the behavior of the procedure may not be defined — often, it will simply crash." [Wooldridge01]

Concurrency & Actions



Transactional Actions

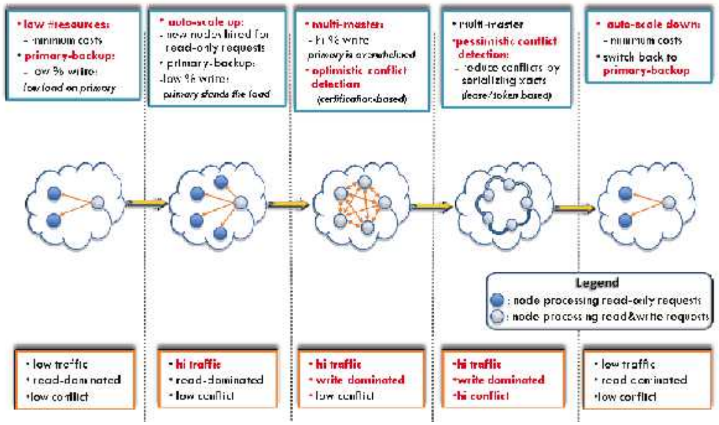
```
tx_monitor do
  @action.execute if @action.applicable?
end
dispatch percepts
```

Scaling vs Consistency

how to ensure (strong) consistency?

- data is distributed and partially replicated among (many) hundreds of servers
- the servers must coordinate to maintain operations on the environment consistent

Autonomic Adaptation



- low #resources:
  - minimum costs
- primary-backup:
  - low % write:
  - low load on primary

- auto-scale up:
  - new nodes hired for read-only requests
- primary-backup:
  - low % write:
  - primary stands the load

- multi-master:
  - hi % write
  - primary is overwhelmed
- optimistic conflict detection (certification-based)

- multi-master
- pessimistic conflict detection:
  - reduce conflicts by serializing xacts (lease/token based)

- auto-scale down:
  - minimum costs
- switch back to primary-backup

Legend

● : node processing read-only requests

○ : node processing read&write requests

- low traffic
- read-dominated
- low conflict

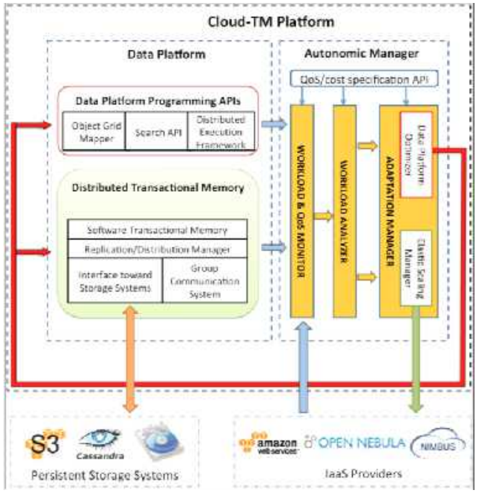
- hi traffic
- read-dominated
- low conflict

- hi traffic
- write dominated
- low conflict

- hi traffic
- write dominated
- hi conflict

- low traffic
- read dominated
- low conflict

Cloud-TM Platform



Cloud-TM Platform

Data Platform

Data Platform Programming APIs

Object Grid Mapper | Search API | Distributed Execution Framework

Distributed Transactional Memory

Software Transactional Memory

Replication/Distribution Manager

Interface toward Storage Systems | Group Communication System

Autonomic Manager

QoS/cost specification API

WORKLOAD & QoS MONITOR

WORKLOAD ANALYZER

ADAPTATION MANAGER

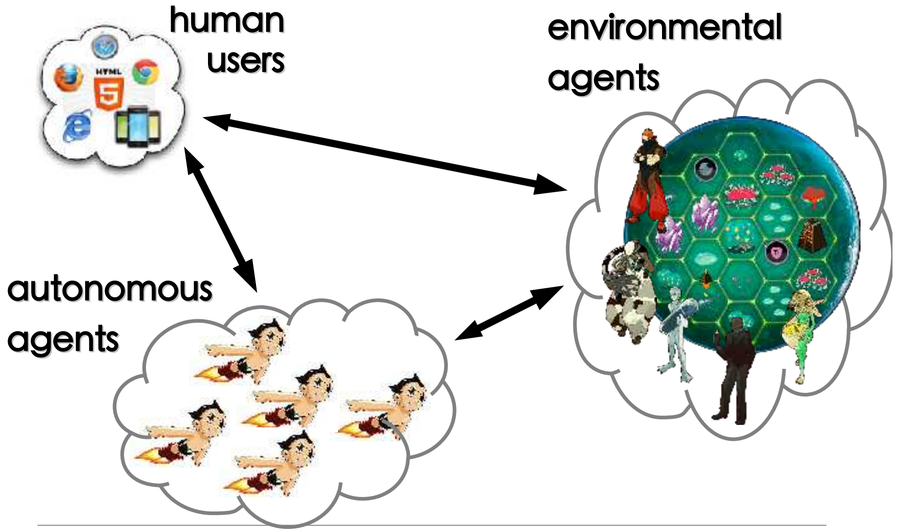
Data Platform Optimizer

Elastic Scaling Manager

Persistent Storage Systems

IaaS Providers

MADMASS Architecture



human users

environmental agents

autonomous agents

Farm Control

- control interface
  - CRUD on groups and agents
  - play/pause/stop
  - benchmarking editor (available soon)
- autonomous agent API
  - custom agent
  - PNP agent (ongoing work)

MADMASS benefits

- open source experimental platform
  - large scale MAS simulations
  - experiments in the cloud (cheap & fast)
- just focus on your research code

The Harvestar too?

- open-source soon
- benefits of madmass
- many research challenges:
  - machine learning (we have logs)
  - bargaining and markets (Farinelli/Tamassia)
  - non-cooperative game theory
  - cooperative game theory (coalition formation)
  - human-agent interaction

Let's team up!

- use the the Harvestar domain to test/develop your research with a real application
- use MADMASS to build your build your own MAS
- we are happy to help!

Behavior Design

Custom Agents ?

Designing bots as *custom agents* for the Harvestar is not trivial

- 4 players, non zero sum
- async: no turns & exogenous events
- non determinism and partial observability
- 4 vs 4 parallel bargaining/ system markets
- coalition formation

Autonomous Agents

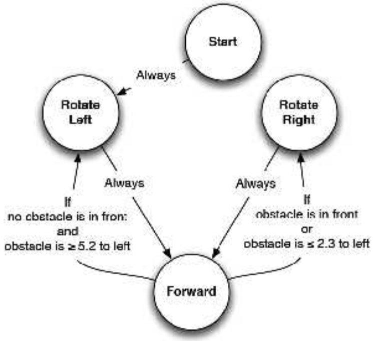
- we would like to provide general purpose tools
- but real-time/multi-agent planners
  - aren't expressive enough to represent all the features of interest
  - are computationally too expensive for real applications

Robot Soccer

Plan Design

- put human intelligence into the process
- tools to represent, analyze and execute plans
  - FSA-based
  - BDI-based
  - Petri Net-based

FSA-Based


Start
Always
Rotate Left
Rotate Right
Always
Always
If no obstacle is in front and obstacle is ≥ 5.2 to left
If obstacle is in front or obstacle is ≤ 2.3 to left
Forward

Rapidly grow in size

e.g., it grows exponentially in the number of actions to model concurrency

BDI

In a Belief, Desire, Intention framework (BDI), agents select behaviors to be executed (intentions) based on their goals (desires), and the current representation of the environments state (beliefs).

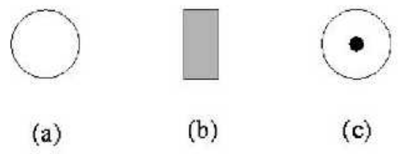
- reduces plan size, but rough cut (mainly reactive plans)
- anyway needs FSA plans (intentions)
- no analysis tool available

Petri Net based

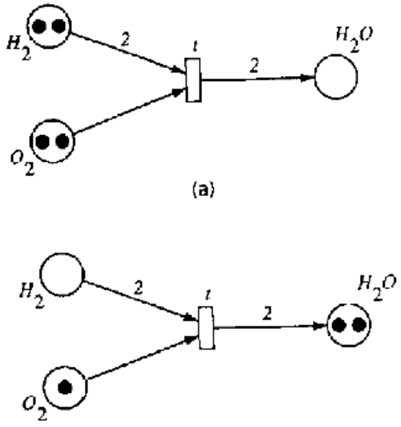
Models of behavior based on Petri Nets (rather than FSA).

- more expressive/exponentially more compact than FSA
- many analysis tools
- yet, complex to design and no general language, just ad hoc models

Petri-Nets



Firing Rule



Petri-Net Plans

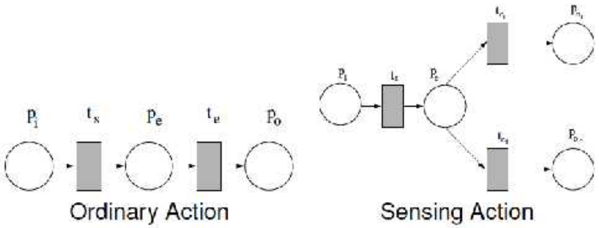
A systematic and methodological approach for MAS Plan Design based on Petri nets that allows for Distributed Execution of multi-agent plans.

Ziparo, Iocchi, Lima, Nardi, Palamara Petri Net Plans: A Framework for Cooperation and Coordination in Multi-Robot Systems. *Journal of Autonomous Agents and Multi-Agent Systems* , 2010.

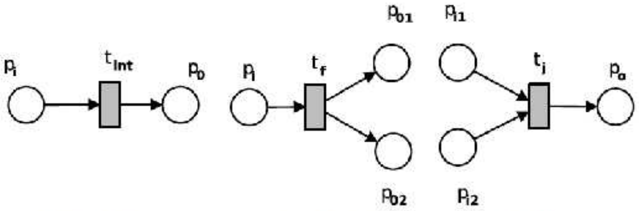
Main Features

- take inspiration from action languages [Rei01]
- concurrent/non-instantaneous actions
- sensing actions [SL93,DINR97]
- coordination & collaborative behaviors **[CL91]**
- plan analysis & distributed execution

Non instantaneous Actions

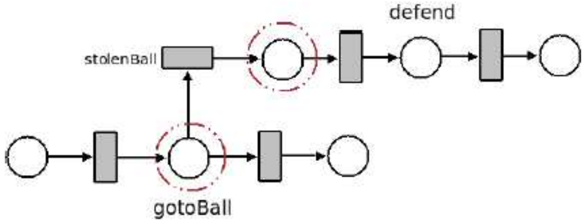


Operators

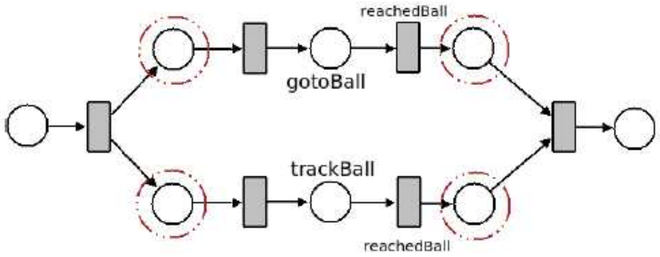


(a) Interrupt (b) Fork (c) Join

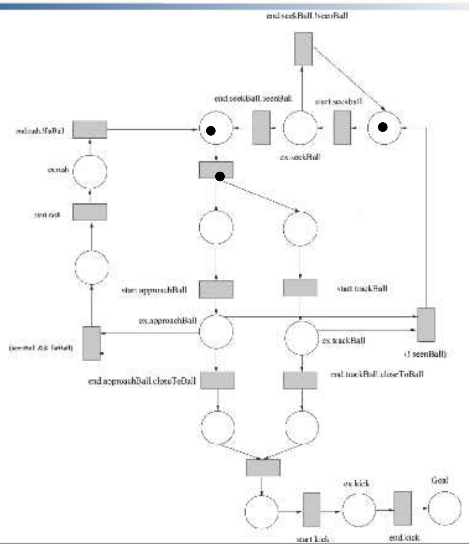
Interrupt



Fork and Join



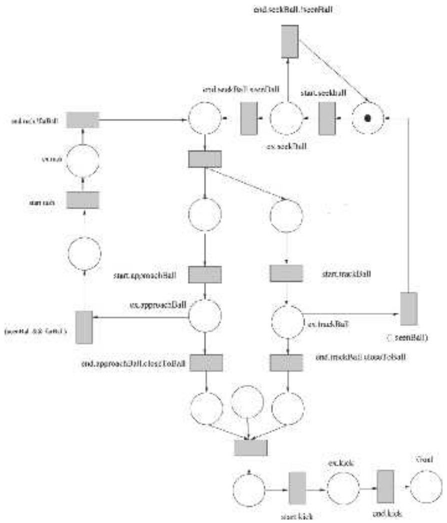
Example



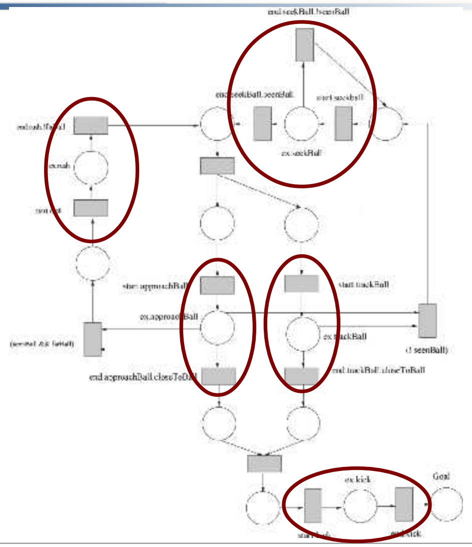
Plan Analysis

- We reduce to standard PN analysis tools that can be directly used on PNPs.
  - safe: thread-wise
  - minimal: no unreachable states
  - effective: no “dead ends”

Analysis Example



Sub-Plans



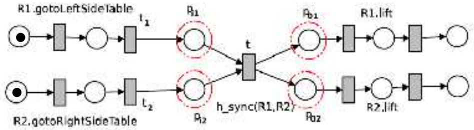
In action: Striker

In action: Goalie

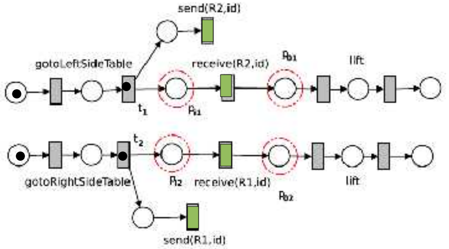
Multi-Agent PNP

- for small teams (2-10 agents)
- centralized design, distributed execution
- based on three operators
  - soft sync
  - hard sync
  - joint commitment

Hard Sync

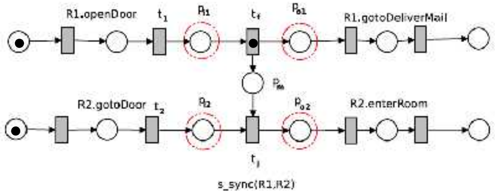


Centralized

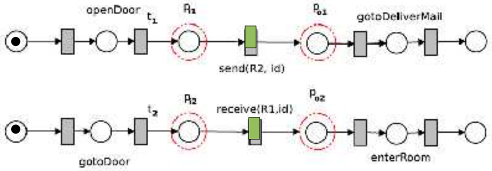


Distributed

Soft Sync

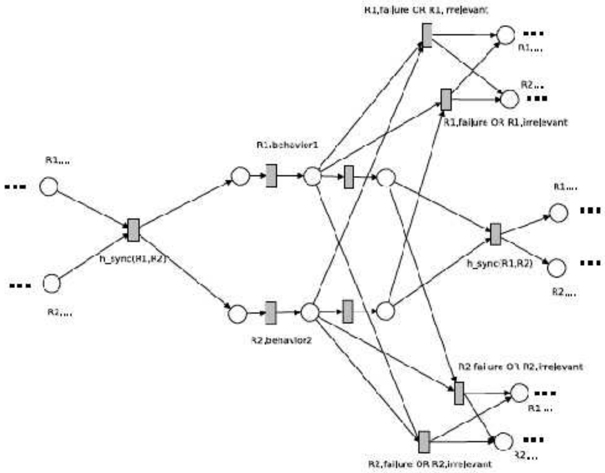


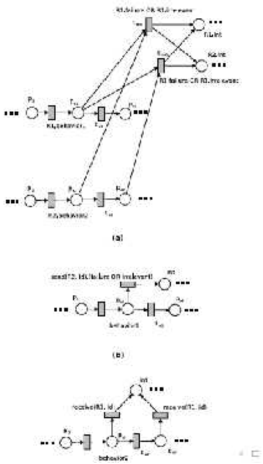
Centralized



Distributed

Joint Intentions [CL91]


R1,failure OR R1, irrelevant
R1,...
R2...
R1,failure OR R1,irrelevant
R1.behavior1
R1...
R1...
h_sync(R1,R2)
h_sync(R1,R2)
R2...
R2...
R2,behavior2
R2 failure OR R2,irrelevant
R1 ...
R2...
R2,failure OR R2,irrelevant



Properties

Property 1. It can be proved that the behavior of the distributed execution “is the same” of the centralized one when robots have access to a reliable communication channel

Property 2. It can be proved that properties verified for the centralized model, hold also for the distributed one.

In action: passing

Best Robotic Demo Award

Int. Conf. Autonomous Agents and Multi Agent Systems 2008

Thanks!