Chemically-Inspired Runtimes for Large Scale Adaptive Computing Platforms

Cédric Tedeschi

Rennes - April 11th, 2017

• 2009-2016 •

Explore chemically-inspired programming to express service coordination and enact it at large scale

A series of works at the crossroads of:

- Autonomic computing
- Chemically-inspired computing
- Workflow management
- Distributed systems

Autonomic Computing (2001-. . .)

Towards Self-organization

[Parashar &Hariri, 2004]

Conceptual research issues and challenges include defining **appropriate abstractions** *and models for* **specifying**, *understanding, controlling, and* **implementing** *autonomic behaviors.*

Autonomic Computing (2001-. . .)

Towards Self-organization

[Parashar &Hariri, 2004]

Conceptual research issues and challenges include defining **appropriate abstractions** *and models for* **specifying**, *understanding, controlling, and* **implementing** *autonomic behaviors.*

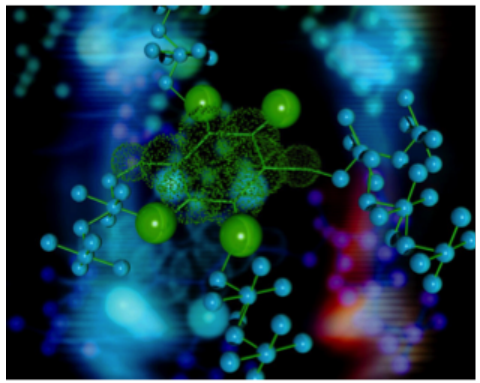
Specification

Study high-level programming models, explore rule-based languages

Implementation

Develop generic runtimes, make them viable at large scale

Chemical computing (1988-. . .)

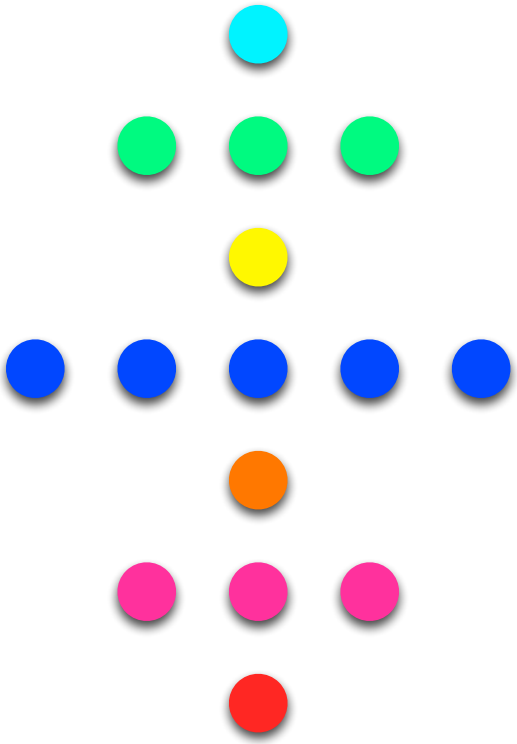


An implicitly parallel rule-based model

- 1988: Gamma
- 1988-1994: Implementation over parallel computers
- 1994: Higher-Order (composition of rules)
- 1998: Structured Gamma
- 2004: Chemically-inspired autonomous systems
- 2006: HOCL
- 2008: Chemically-inspired service orchestration

Workflows

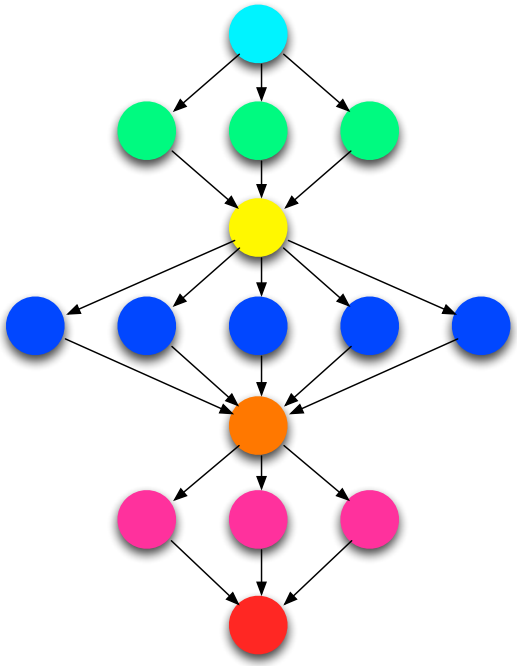
Temporal, loosely-coupled composition of services



- Centralized management
- Static reconfiguration

Workflows

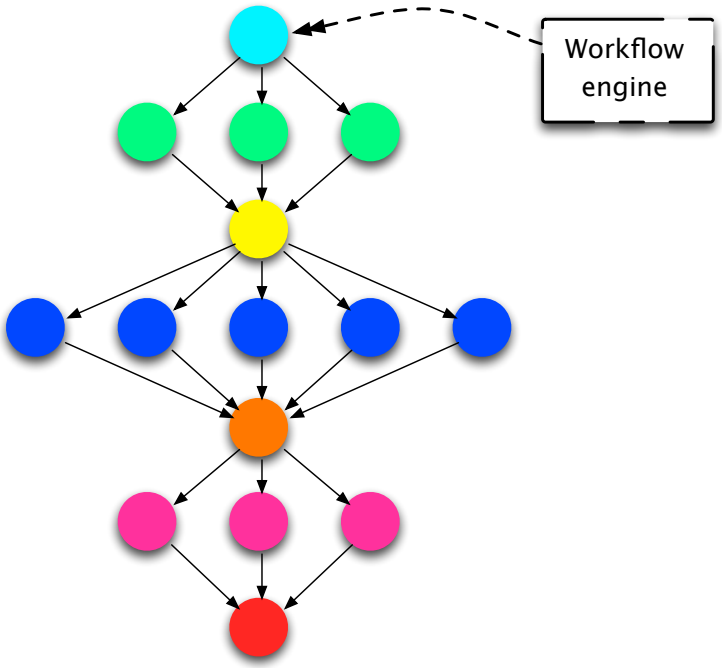
Temporal, loosely-coupled composition of services



- Centralized management
- Static reconfiguration

Workflows

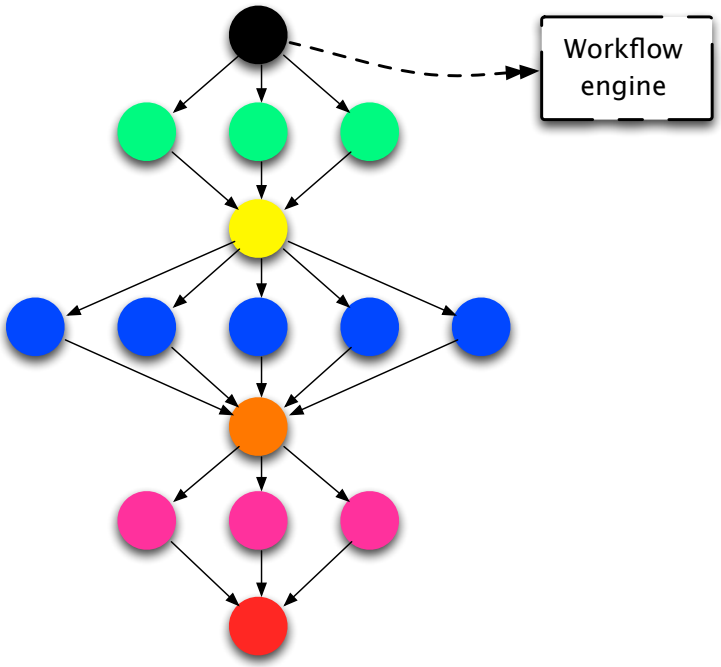
Temporal, loosely-coupled composition of services



- Centralized management
- Static reconfiguration

Workflows

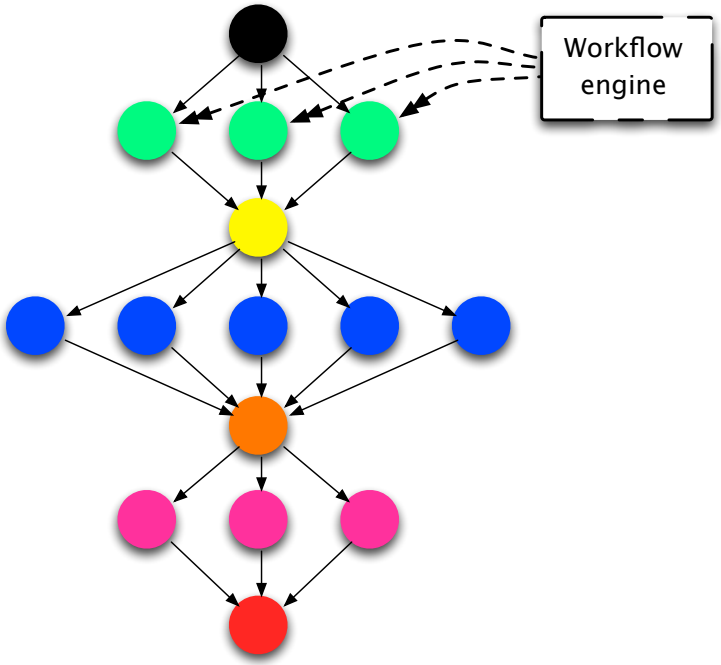
Temporal, loosely-coupled composition of services



- Centralized management
- Static reconfiguration

Workflows

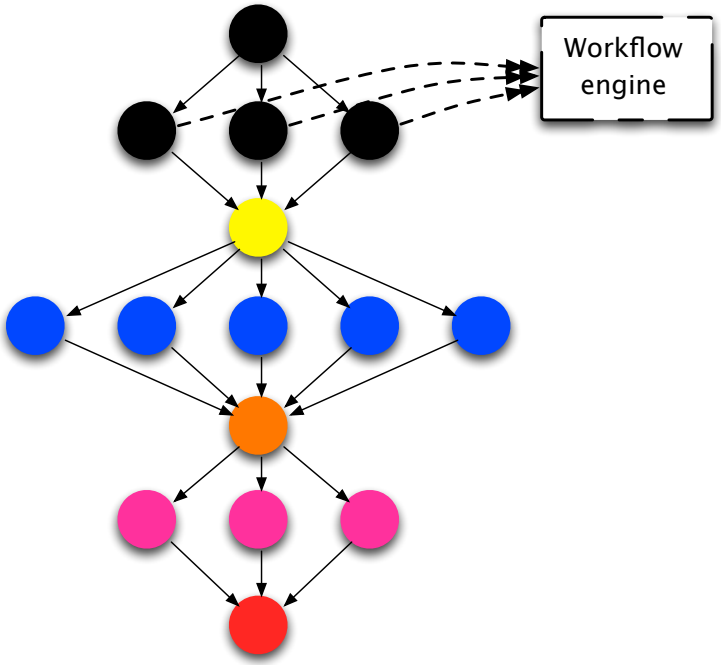
Temporal, loosely-coupled composition of services



- Centralized management
- Static reconfiguration

Workflows

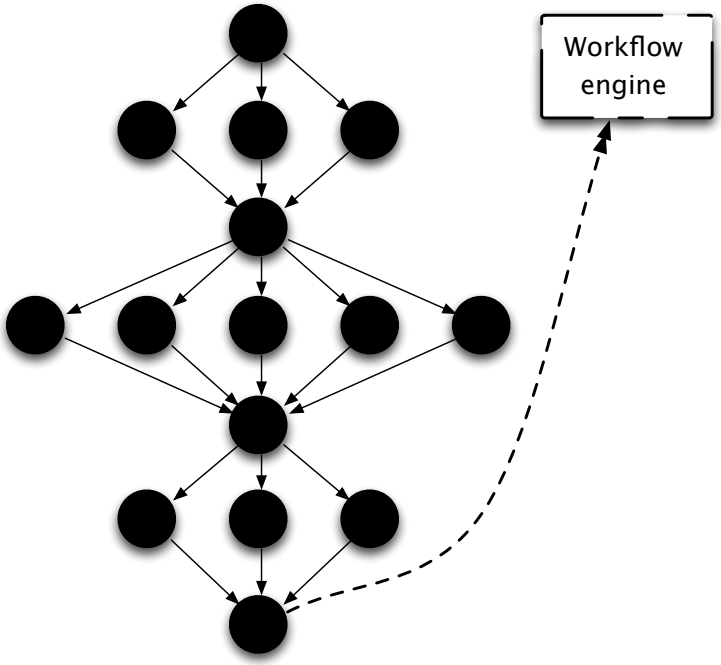
Temporal, loosely-coupled composition of services



- Centralized management
- Static reconfiguration

Workflows

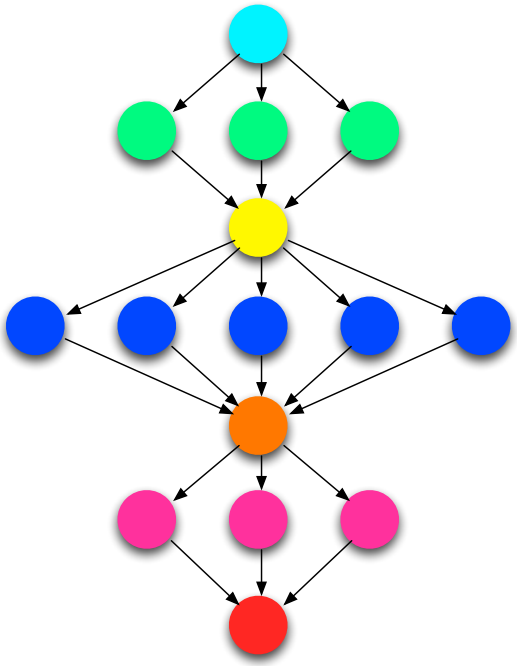
Temporal, loosely-coupled composition of services



- Centralized management
- Static reconfiguration

Workflows

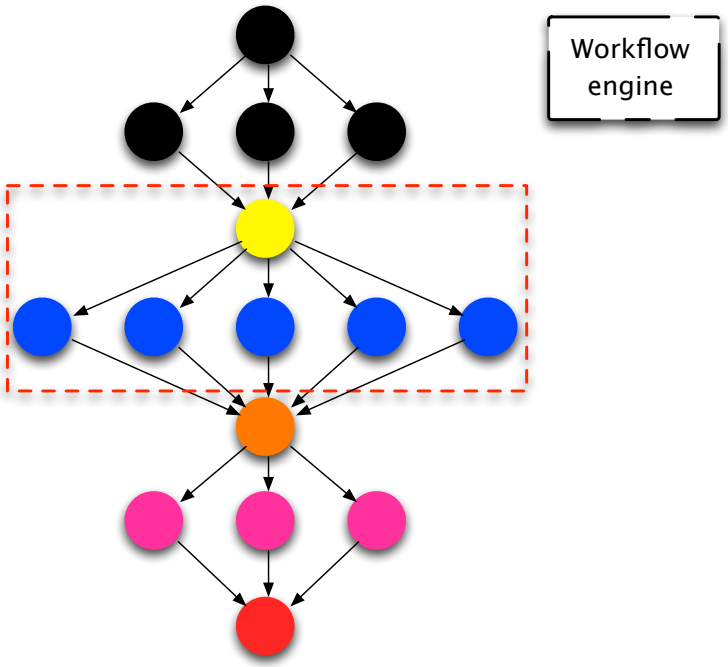
Temporal, loosely-coupled composition of services



- Centralized management
- Static reconfiguration

Workflows

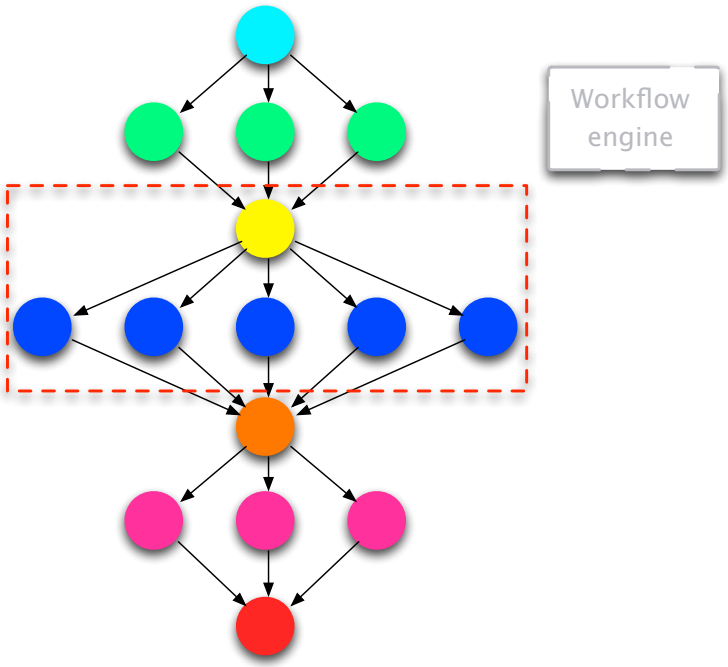
Temporal, loosely-coupled composition of services



- Centralized management
- Static reconfiguration

Workflows

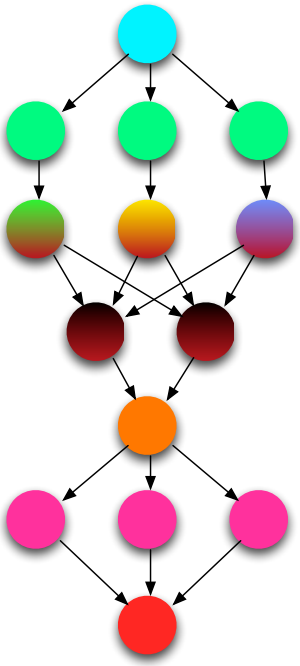
Temporal, loosely-coupled composition of services



- Centralized management
- Static reconfiguration

Workflows

Temporal, loosely-coupled composition of services

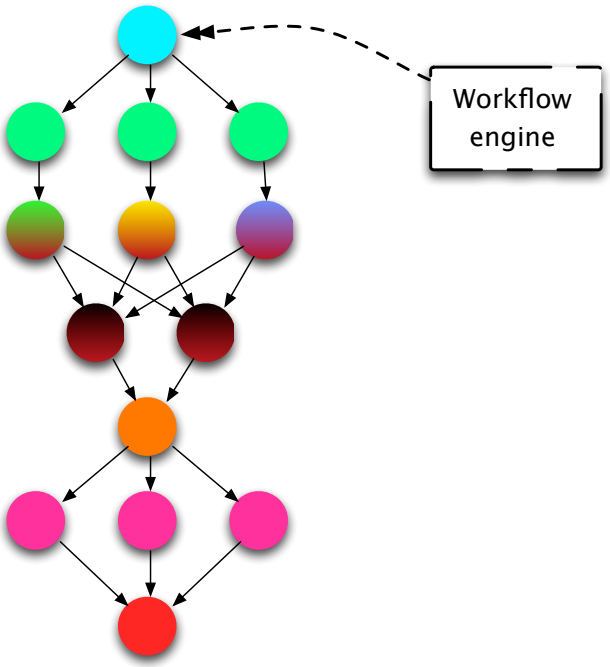


Workflow engine

- Centralized management
- Static reconfiguration

Workflows

Temporal, loosely-coupled composition of services



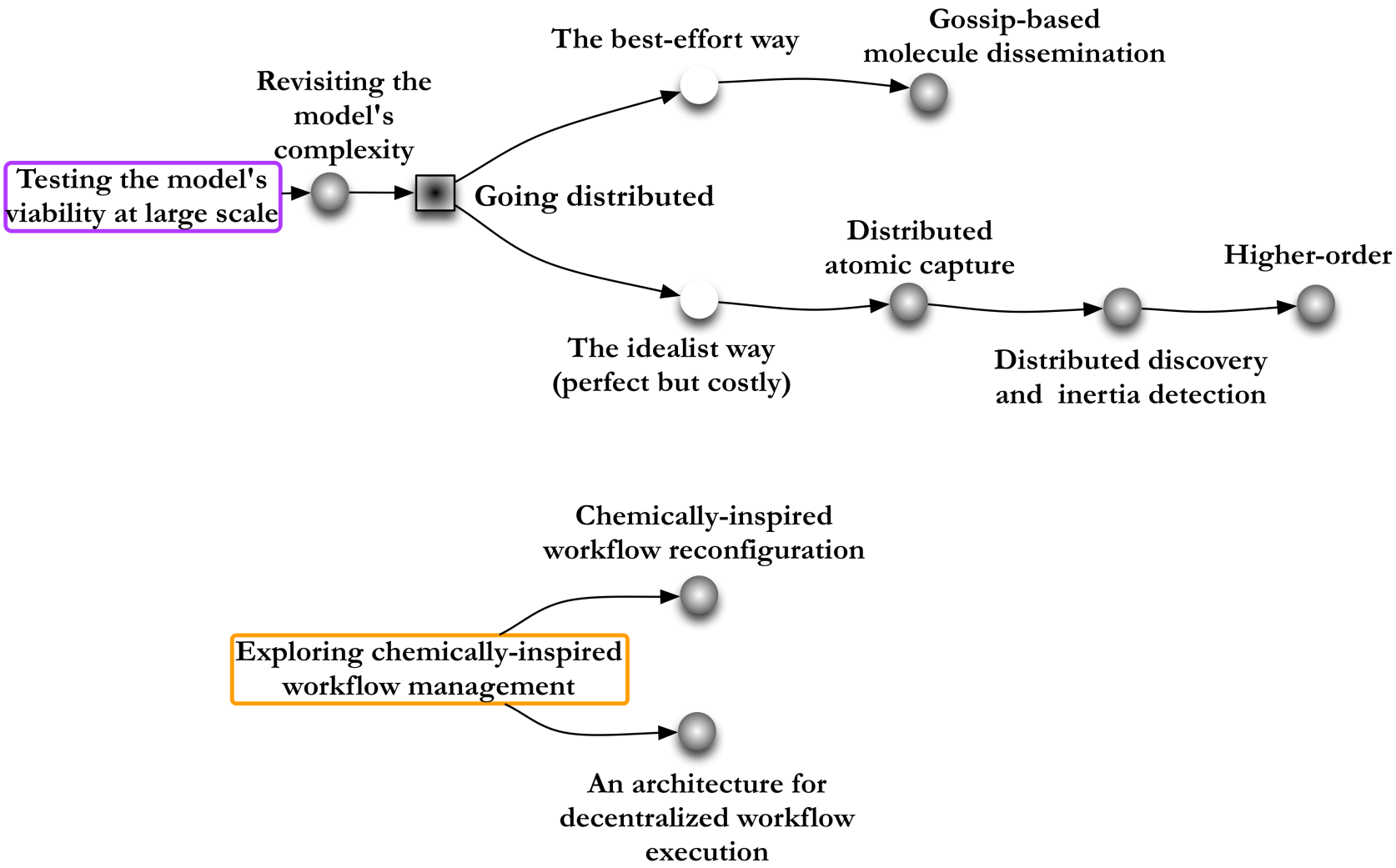
- Centralized management
- Static reconfiguration

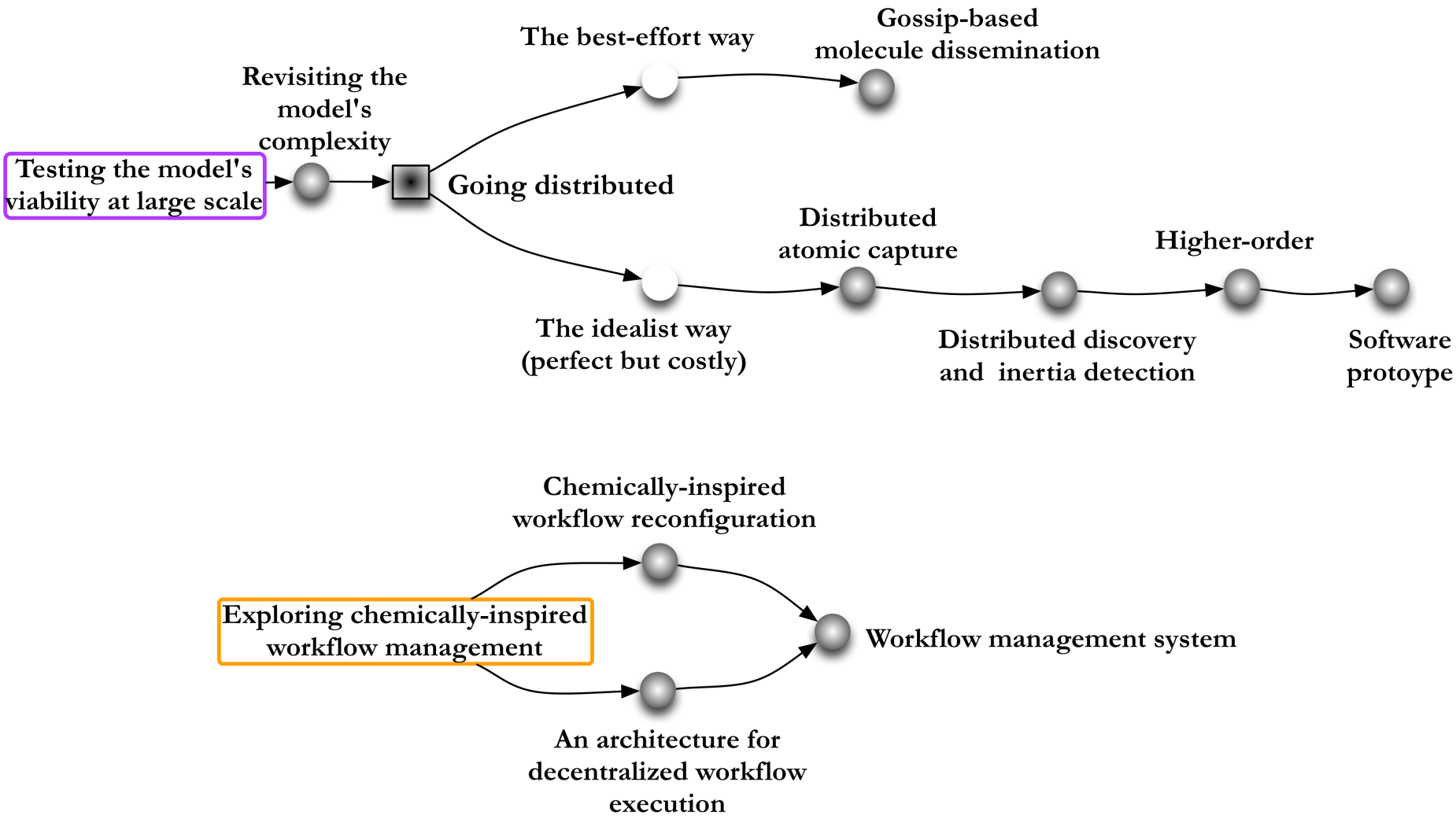
• Open problems (2009) •

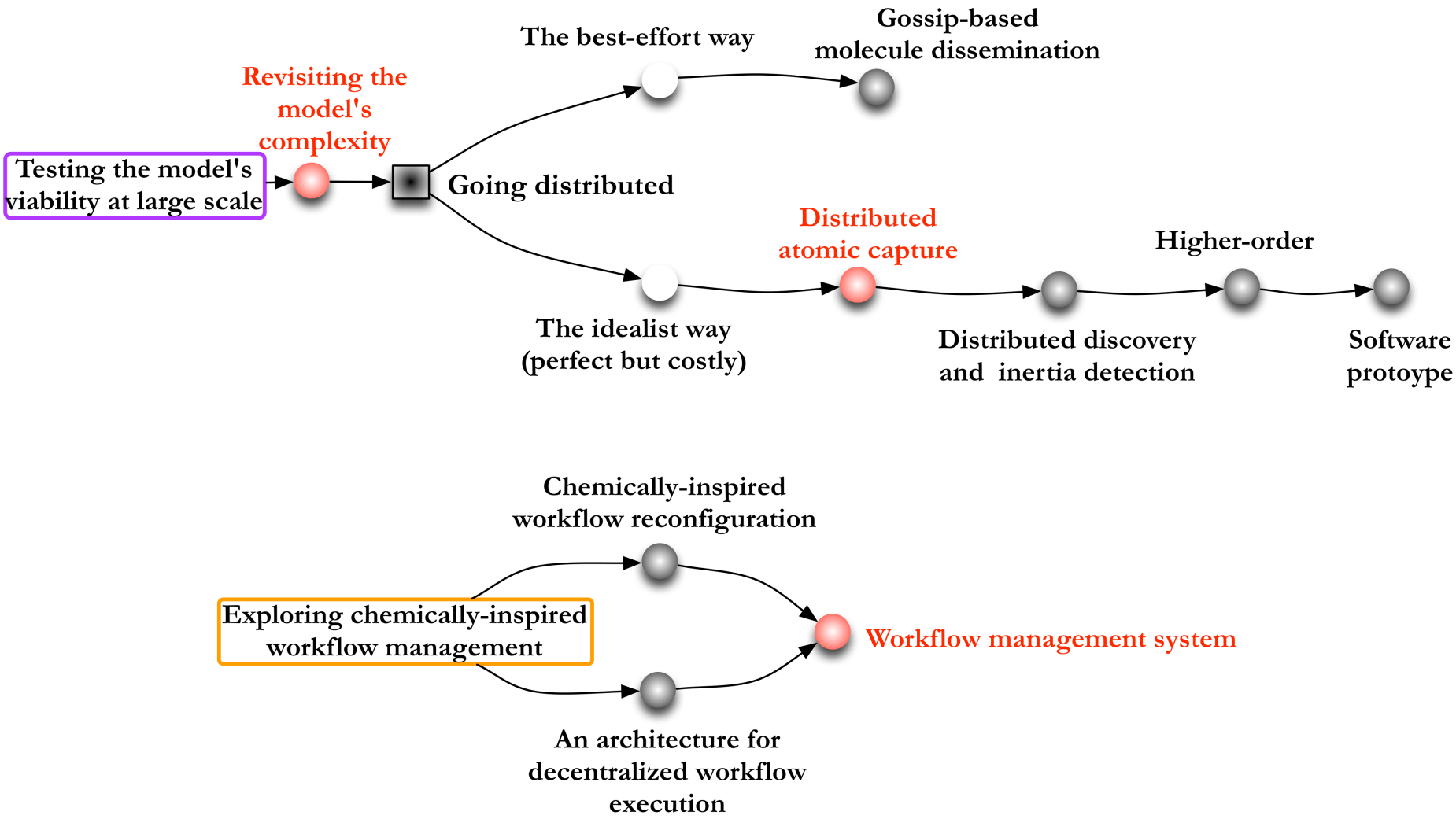
- How can the chemical model help specify coordination and adaptation?
- How to decentralise the coordination of workflow execution?
- How to reconfigure a workflow's structure on-the-fly ?
- How can the chemical model be made viable at large scale?

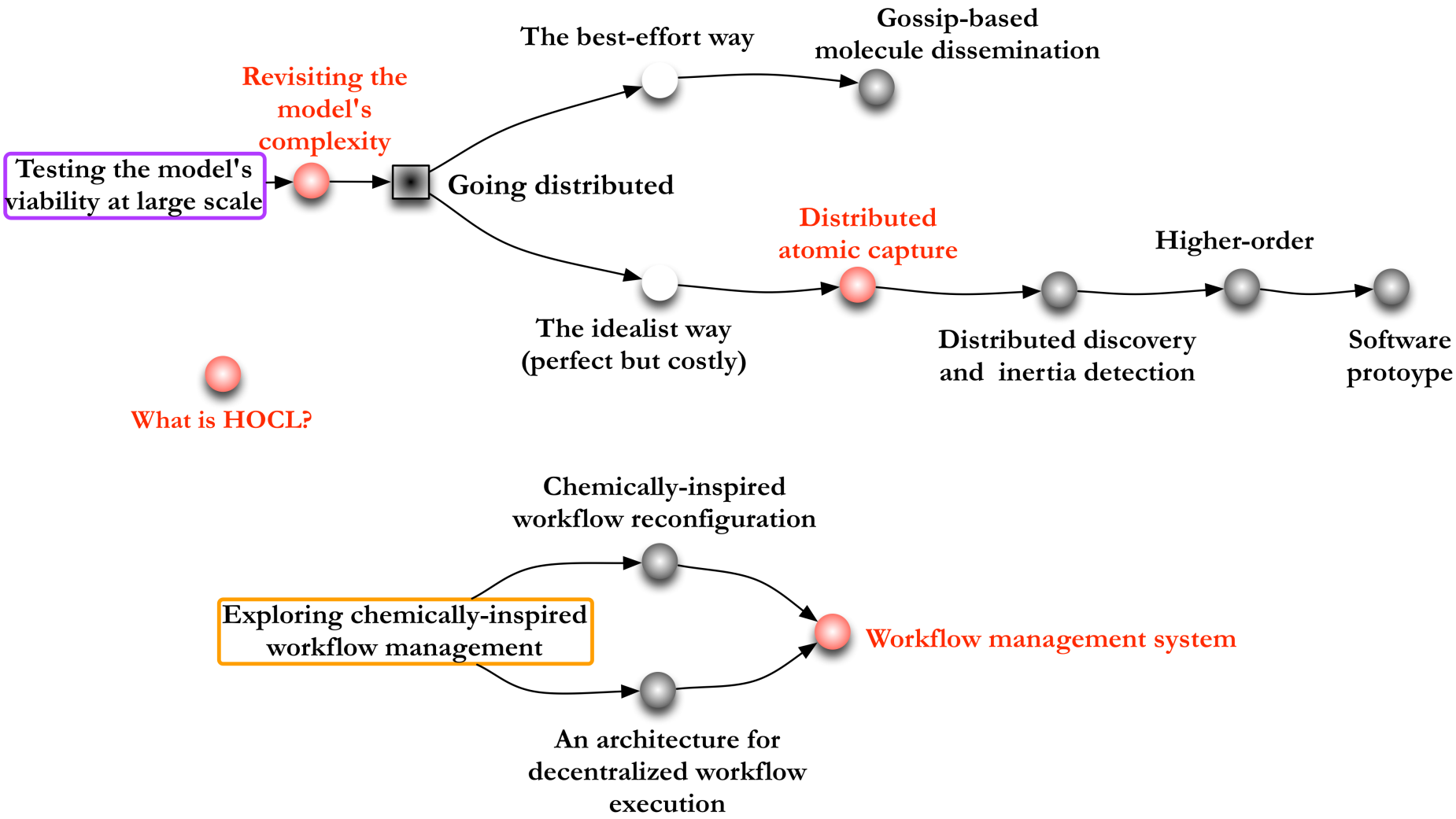
Testing the model's viability at large scale

Exploring chemically-inspired workflow management


Testing the model's viability at large scale
Revisiting the model's complexity
Going distributed
The best-effort way
Gossip-based molecule dissemination
The idealist way (perfect but costly)
Distributed atomic capture
Distributed discovery and inertia detection
Higher-order
Exploring chemically-inspired workflow management
Chemically-inspired workflow reconfiguration
An architecture for decentralized workflow execution


Testing the model's viability at large scale
Revisiting the model's complexity
Going distributed
The best-effort way
Gossip-based molecule dissemination
The idealist way (perfect but costly)
Distributed atomic capture
Distributed discovery and inertia detection
Higher-order
Software protoype
Exploring chemically-inspired workflow management
Chemically-inspired workflow reconfiguration
An architecture for decentralized workflow execution
Workflow management system


Testing the model's viability at large scale
Revisiting the model's complexity
Going distributed
The best-effort way
Gossip-based molecule dissemination
The idealist way (perfect but costly)
Distributed atomic capture
Distributed discovery and inertia detection
Higher-order
Software protoype
Exploring chemically-inspired workflow management
Chemically-inspired workflow reconfiguration
An architecture for decentralized workflow execution
Workflow management system

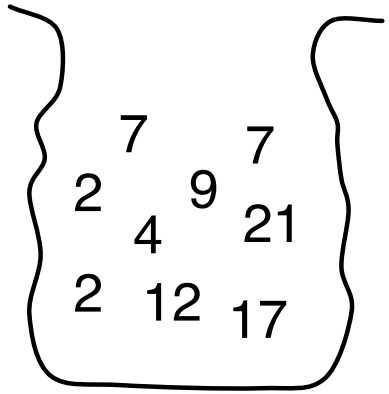

The best-effort way
Gossip-based
molecule dissemination
Revisiting the
model's
complexity
Testing the model's
viability at large scale
Going distributed
Distributed
atomic capture
Higher-order
The idealist way
(perfect but costly)
Distributed discovery
and inertia detection
Software
protoype
What is HOCL?
Chemically-inspired
workflow reconfiguration
Exploring chemically-inspired
workflow management
Workflow management system
An architecture for
decentralized workflow
execution

Part 0

What is HOCL?

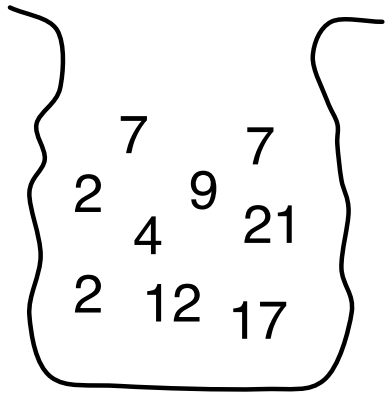
HOCL programming (a.k.a. the usual getMax example)

```
int getMax (int tab [], int size) {
    int max = tab[0];
    for (int i = 1; i < size; i++) {
        if (tab[i] > max)
            max = tab[i];
    }
    return(max);
}
```



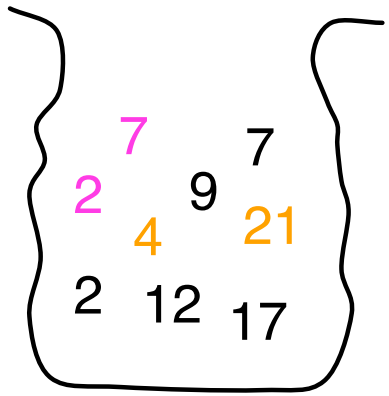
HOCL programming (a.k.a. the usual getMax example)

```
int getMax (int tab [], int size) {
    int max = tab[0];
    for (int i = 1; i < size; i++) {
        if (tab[i] > max)
            max = tab[i];
    }
    return(max);
}
```



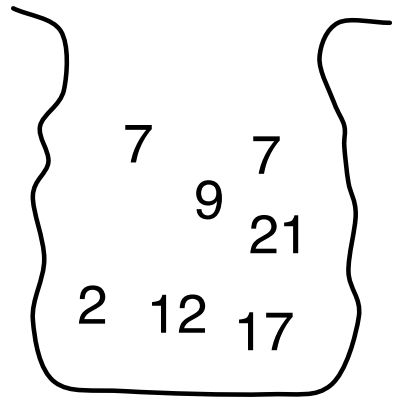

```
replace x, y  by x  if x >= y
```

HOCL programming (a.k.a. the usual getMax example)



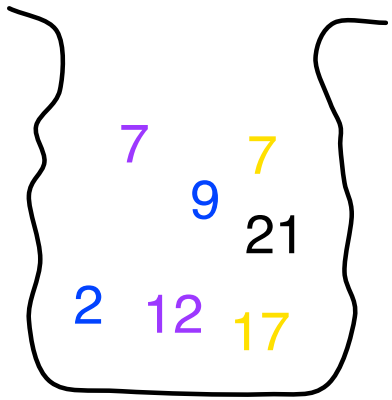

```
replace x, y  by x  if x >= y
```

HOCL programming (a.k.a. the usual getMax example)



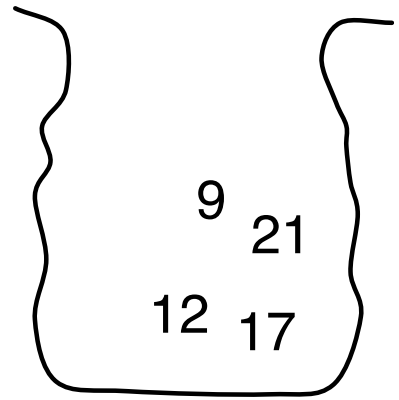

```
replace x, y  by x  if x >= y
```

HOCL programming (a.k.a. the usual getMax example)



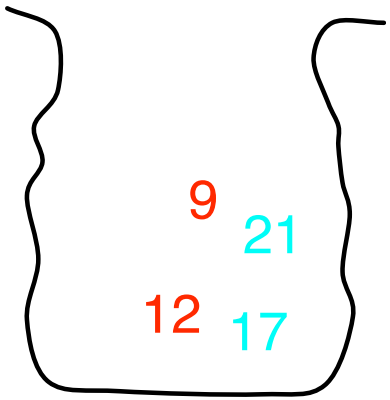

```
replace x, y  by x  if x >= y
```

HOCL programming (a.k.a. the usual getMax example)



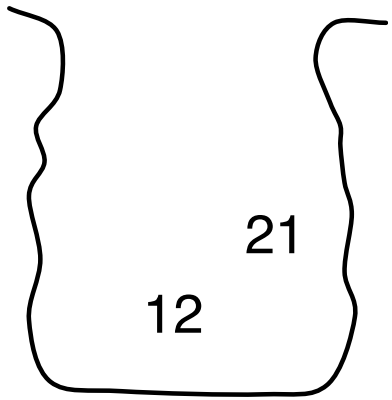

```
replace x, y  by x  if x >= y
```

HOCL programming (a.k.a. the usual getMax example)



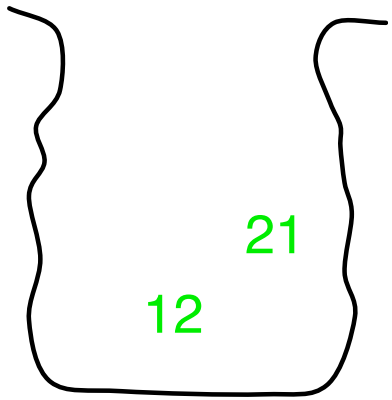

```
replace x, y  by x  if x >= y
```

HOCL programming (a.k.a. the usual getMax example)



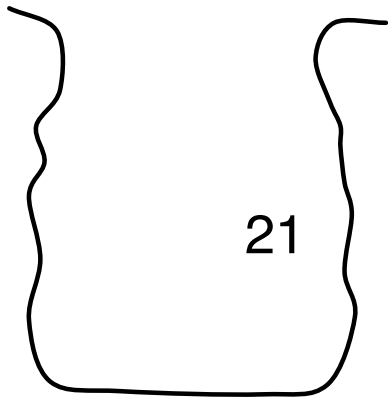

```
replace x, y  by x  if x >= y
```

HOCL programming (a.k.a. the usual getMax example)




```
replace x, y  by x  if x >= y
```

HOCL programming (a.k.a. the usual getMax example)




```
replace x, y  by x  if x >= y
```

HOCL execution model

The basic chemical analogy

- Data are *molecules floating around in a solution*
- When they meet, they react
  - Reactants are consumed
  - New molecules are produced

Implicit parallelism / non determinism

Any enabled reaction can be triggered provided:

- the atomic capture (molecules are consumed only once)
- liveness (to reach termination or *inertia*)

HOCL execution model

The basic chemical analogy

- Data are *molecules floating around in a solution*
- When they meet, they react
  - Reactants are consumed
  - New molecules are produced

Implicit parallelism / non determinism

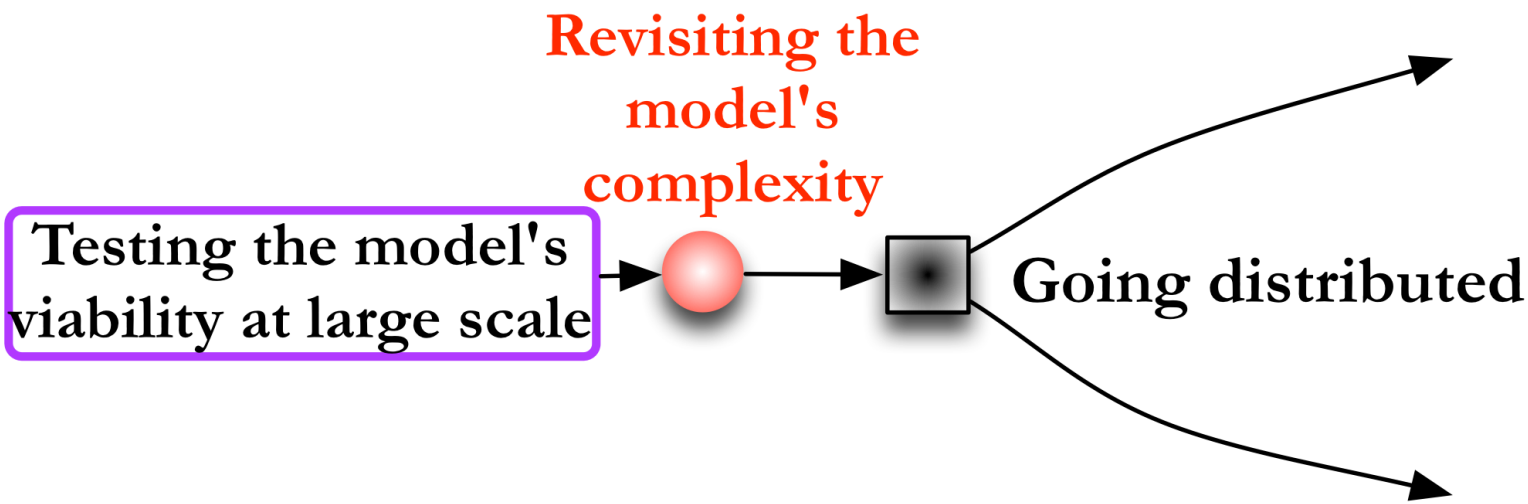
Any enabled reaction can be triggered provided:

- the atomic capture (molecules are consumed only once)
- liveness (to reach termination or *inertia*)

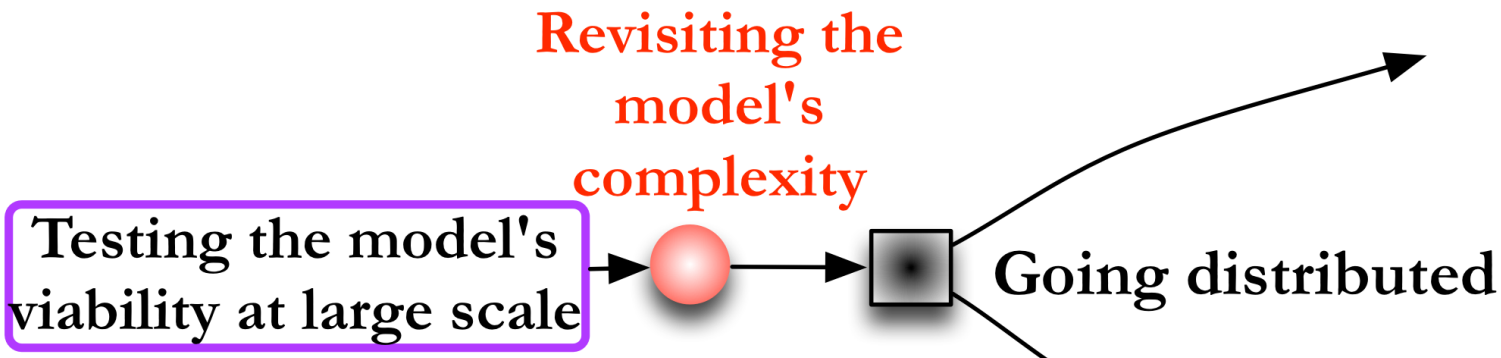
HOCL programming model

- Data left unstructured in a bag / multiset
- Rewriting rules to be applied on the multiset
- Sequentiality through solution nesting
- Higher-order: rules applying on rules

Part I


Revisiting the model's complexity
Testing the model's viability at large scale
Going distributed

Part I



Joint work with
- Matthieu Perrin (M2 ENS Rennes)
- Marin Bertier (INSA Rennes)

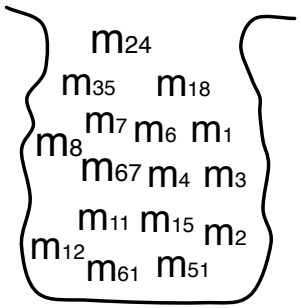
The reactants search problem

[Le Métayer (1994)]

Because of the combinatorial explosion imposed by its semantics, it is difficult to reach a decent level of efficiency in any general implementation of the language.

- *Given a rule, find a set of molecules satisfying its condition*
- A kind of CSP problem
- The brute-force approach answers in $O(n^k)$, provided $n \gg k$.

```
replace  x1, x2, ..., xk  by  f(x1, x2, ..., xk)
                          if  C(x1, x2, ..., xk)
```



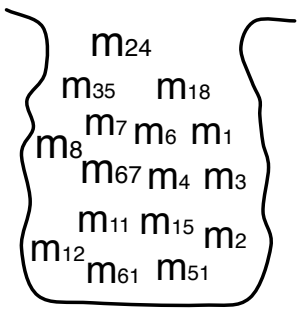
The reactants search problem

[Le Métayer (1994)]

Because of the combinatorial explosion imposed by its semantics, it is difficult to reach a decent level of efficiency in any general implementation of the language.

- *Given a rule, find a set of molecules satisfying its condition*
- A kind of CSP problem
- The brute-force approach answers in $O(n^k)$, provided $n \gg k$.

```
replace x1, x2, ..., xk by f(x1, x2, ..., xk)
                    if C(x1, x2, ..., xk)
```

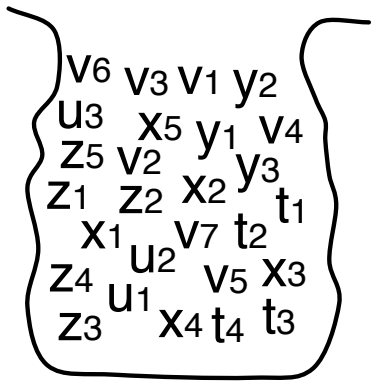


Can we do better?

Can we better characterize the complexity?

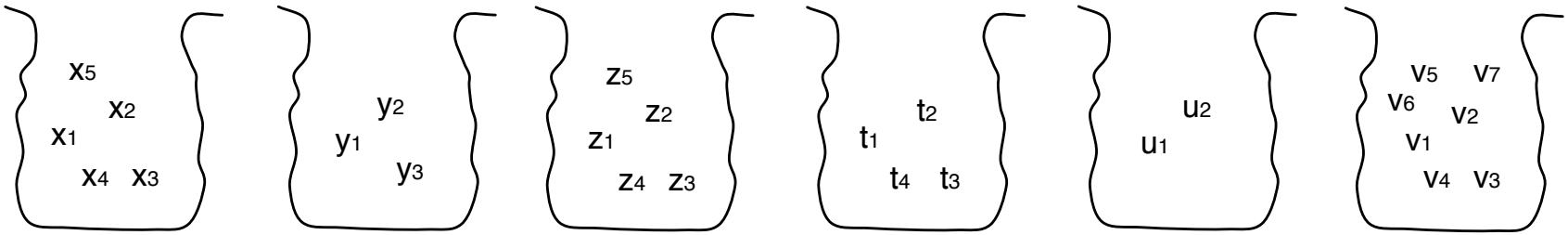
Finding reactants (with no condition)

```
replace x,y,z,t,u,v by f(x,y,z)
```



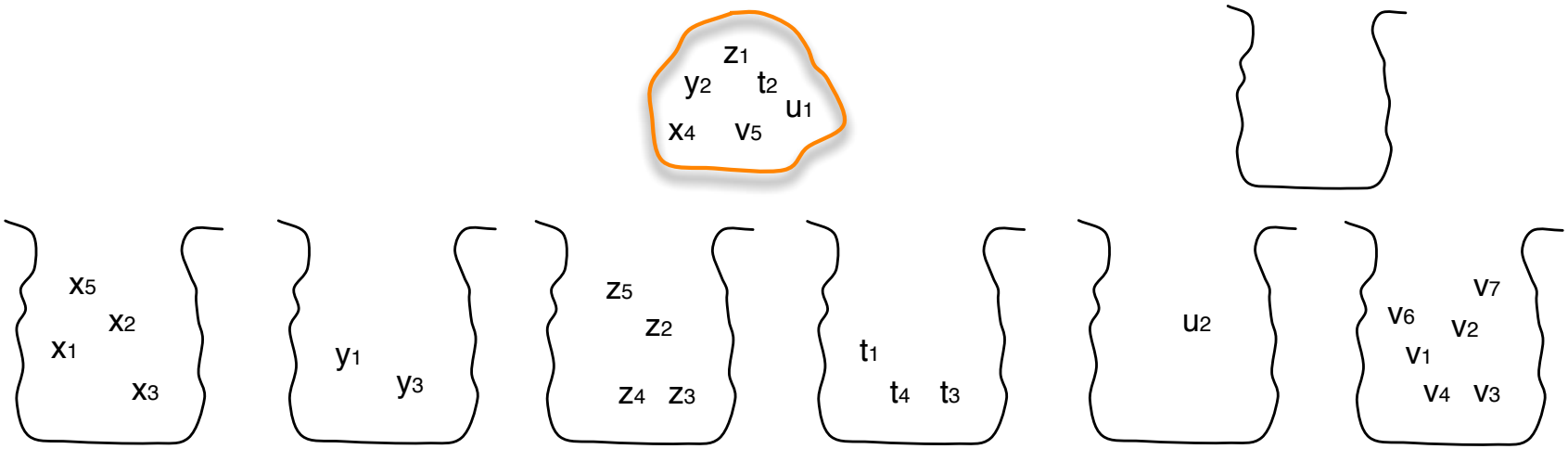
Finding reactants (with no condition)

```
replace x,y,z,t,u,v by f(x,y,z)
```



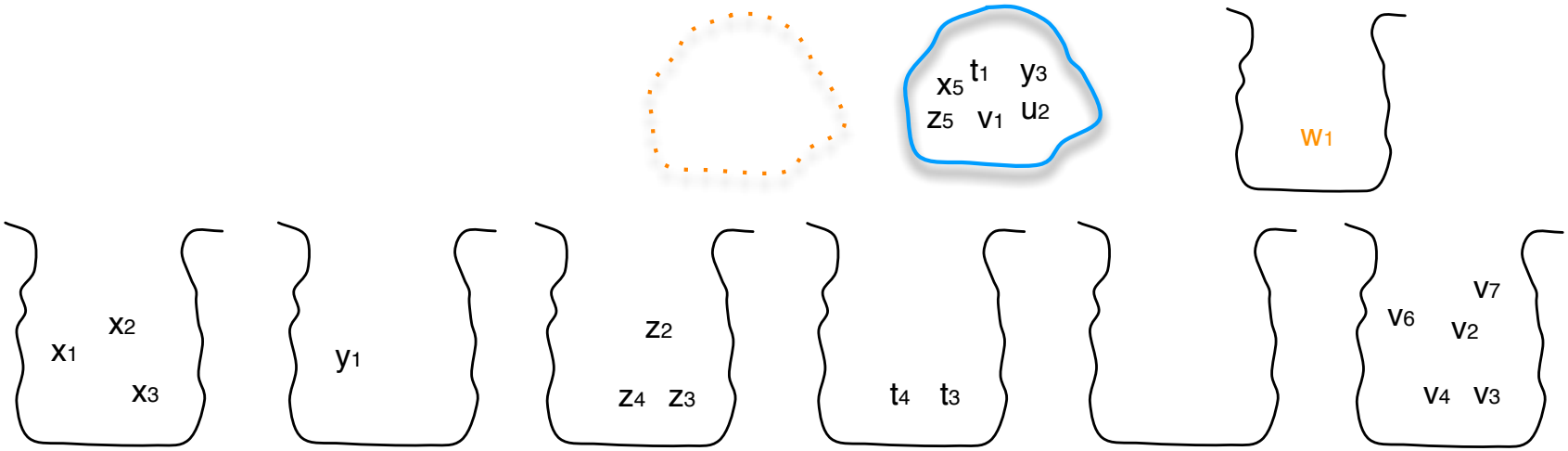
Finding reactants (with no condition)

```
replace x,y,z,t,u,v by f(x,y,z)
```



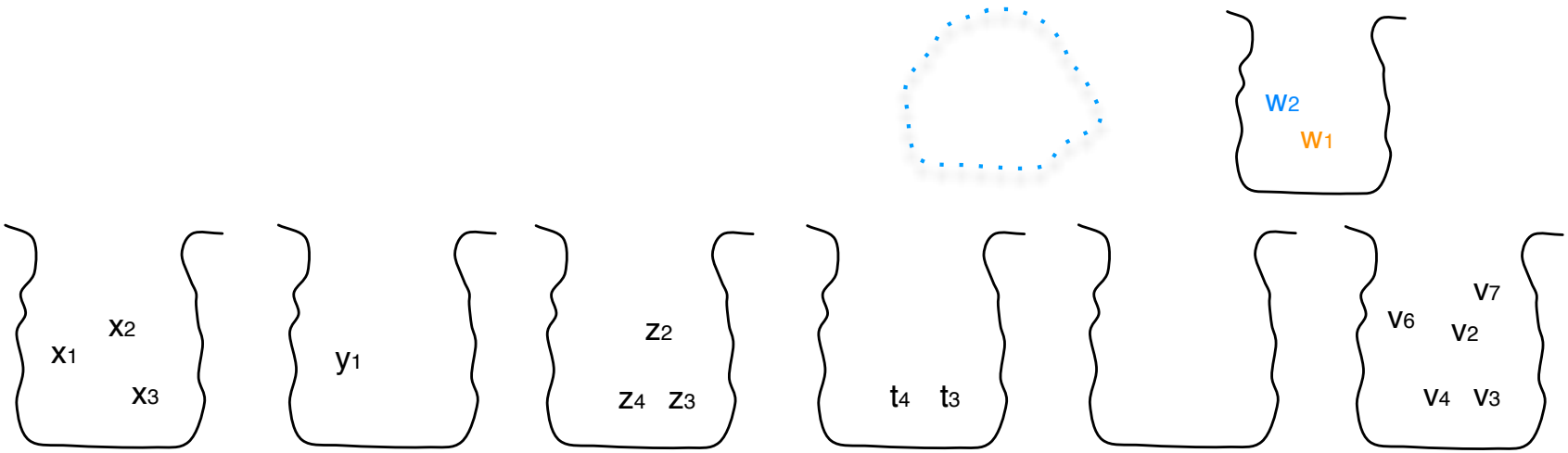
Finding reactants (with no condition)

```
replace x,y,z,t,u,v by f(x,y,z)
```



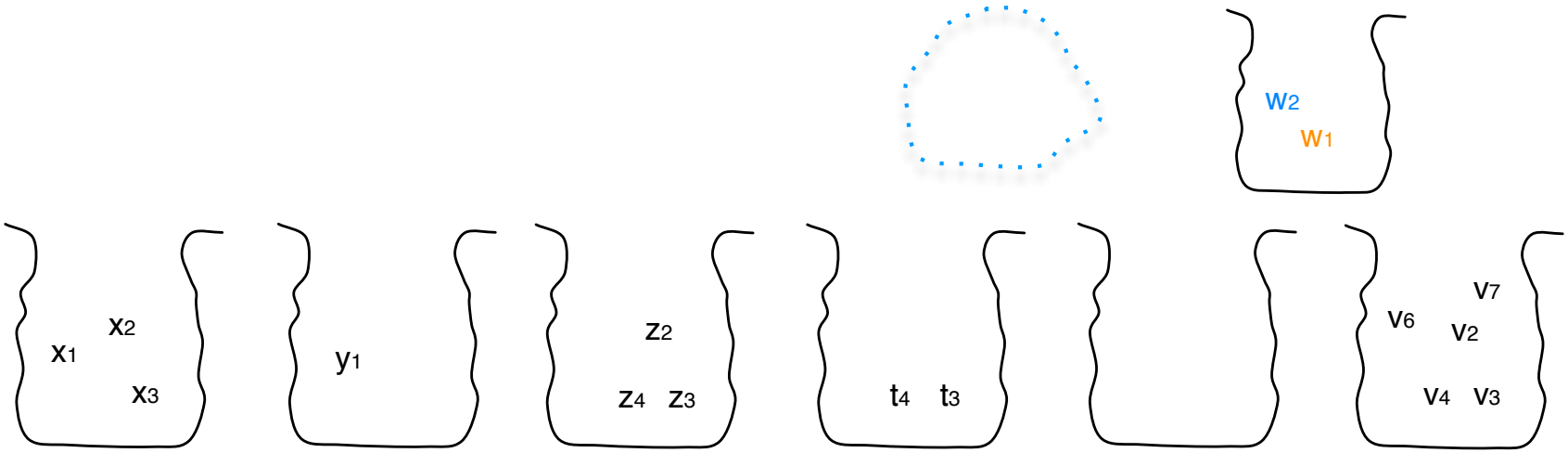
Finding reactants (with no condition)

```
replace x,y,z,t,u,v by f(x,y,z)
```



Finding reactants (with no condition)

```
replace x,y,z,t,u,v by f(x,y,z)
```



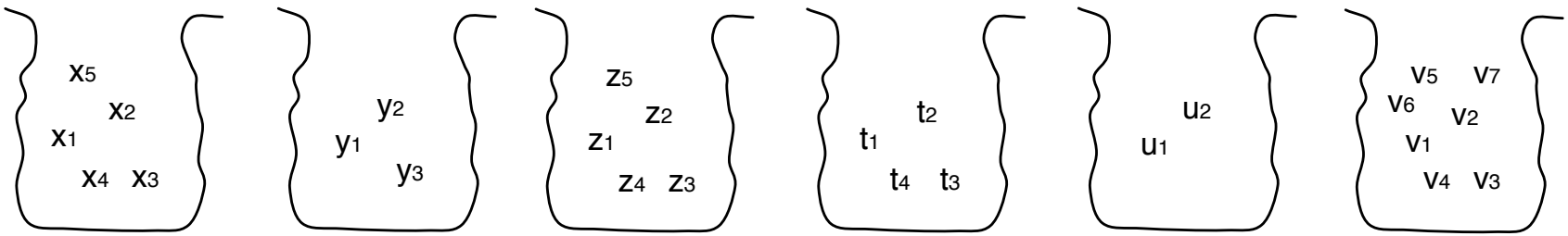
When there is no condition, detecting inertia is linear.

Finding reactants (with a simple condition)

replace x,y,z,t,u,v **by** f(x,y,z) **if** c_1(x,y)

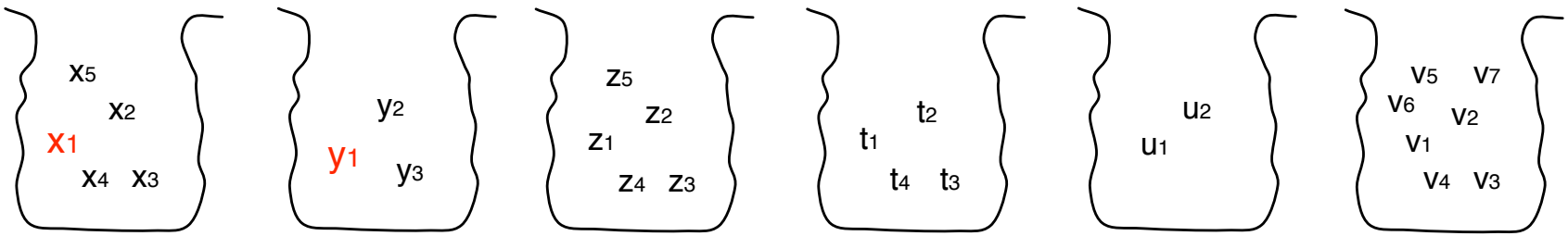
Finding reactants (with a simple condition)

replace x,y,z,t,u,v **by** f(x,y,z) **if** c_1(x,y)



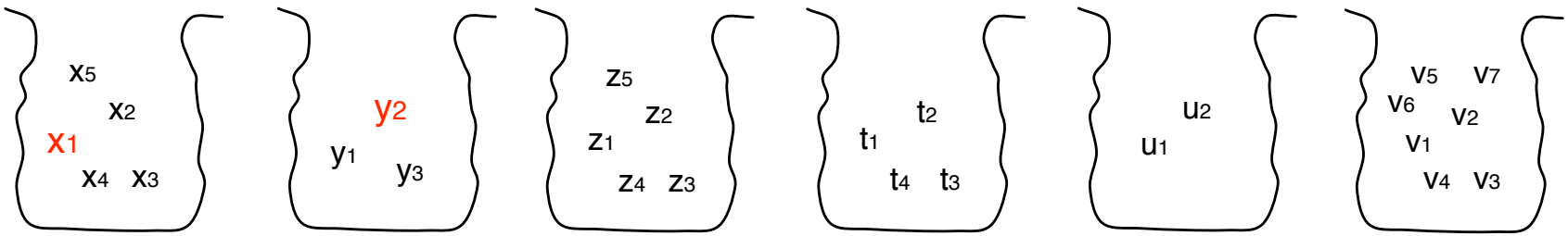
Finding reactants (with a simple condition)

```
replace x,y,z,t,u,v by f(x,y,z) if c1(x,y)
```



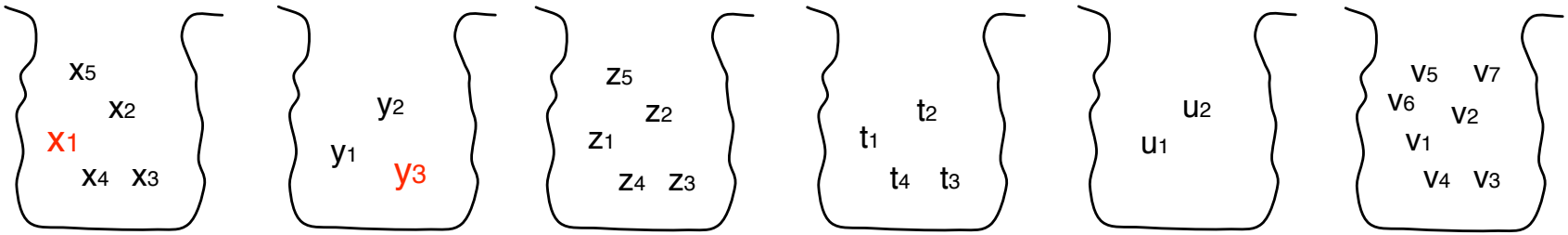
Finding reactants (with a simple condition)

replace x,y,z,t,u,v **by** f(x,y,z) **if** c_1(x,y)



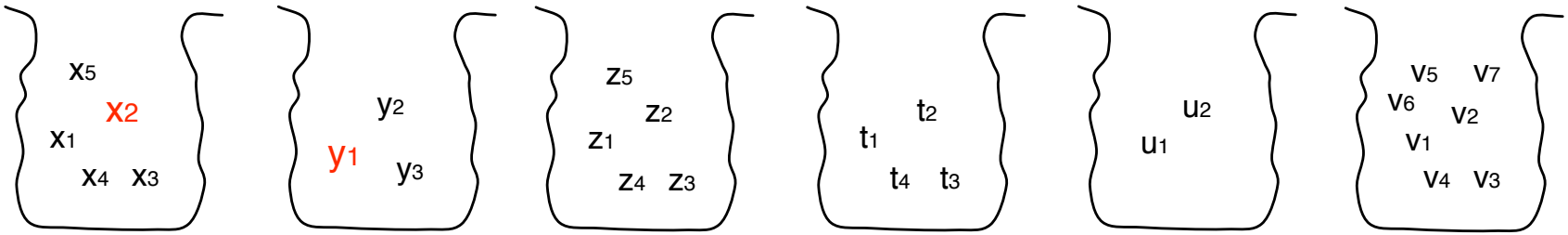
Finding reactants (with a simple condition)

```
replace x,y,z,t,u,v by f(x,y,z) if c1(x,y)
```



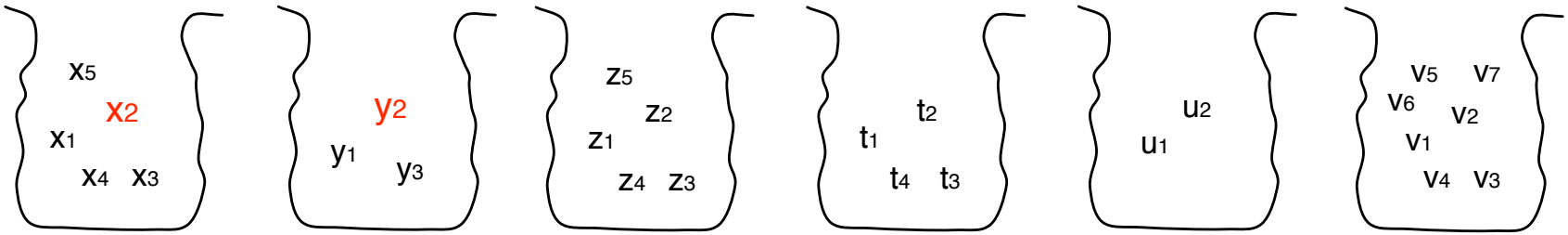
Finding reactants (with a simple condition)

```
replace x,y,z,t,u,v by f(x,y,z) if c1(x,y)
```



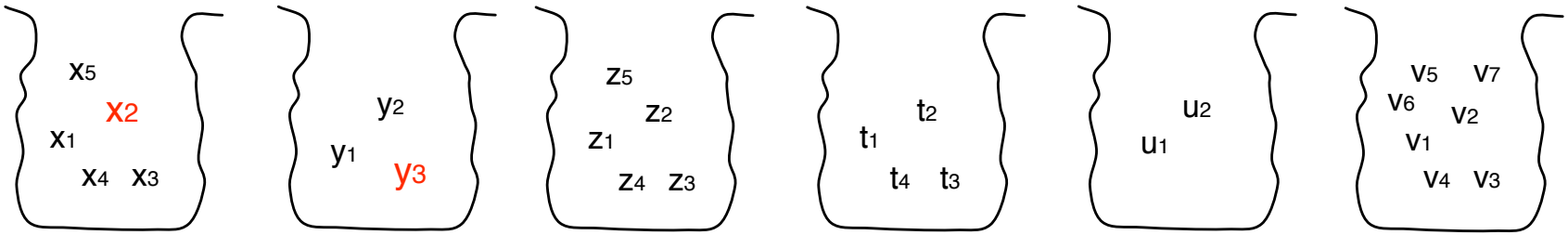
Finding reactants (with a simple condition)

```
replace x,y,z,t,u,v by f(x,y,z) if c1(x,y)
```



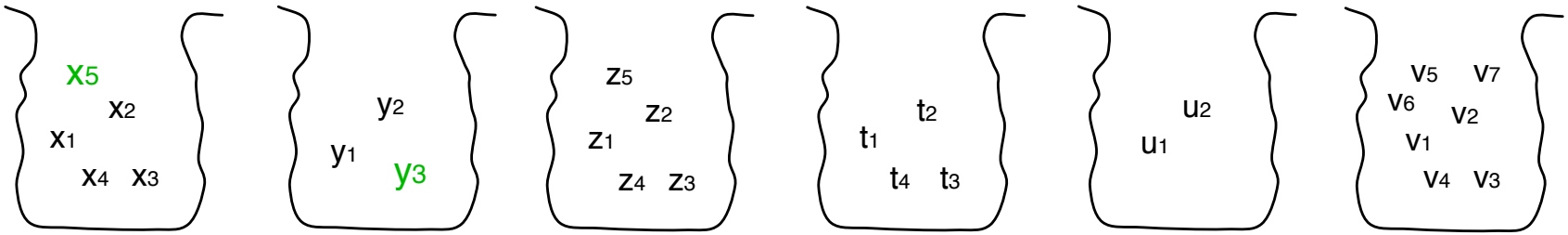
Finding reactants (with a simple condition)

```
replace x,y,z,t,u,v by f(x,y,z) if c1(x,y)
```



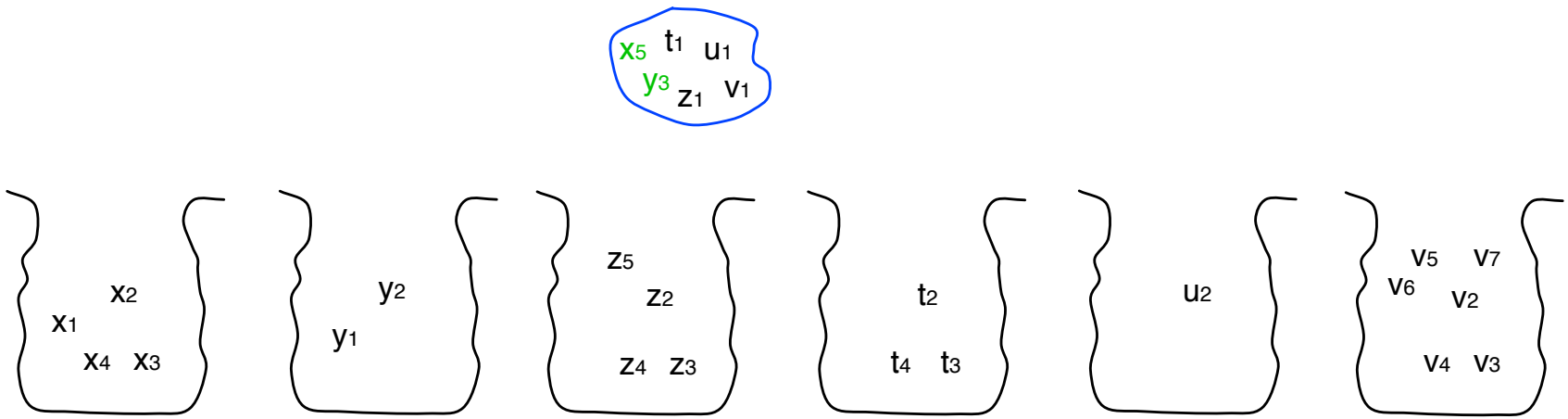
Finding reactants (with a simple condition)

```
replace x,y,z,t,u,v by f(x,y,z) if c1(x,y)
```



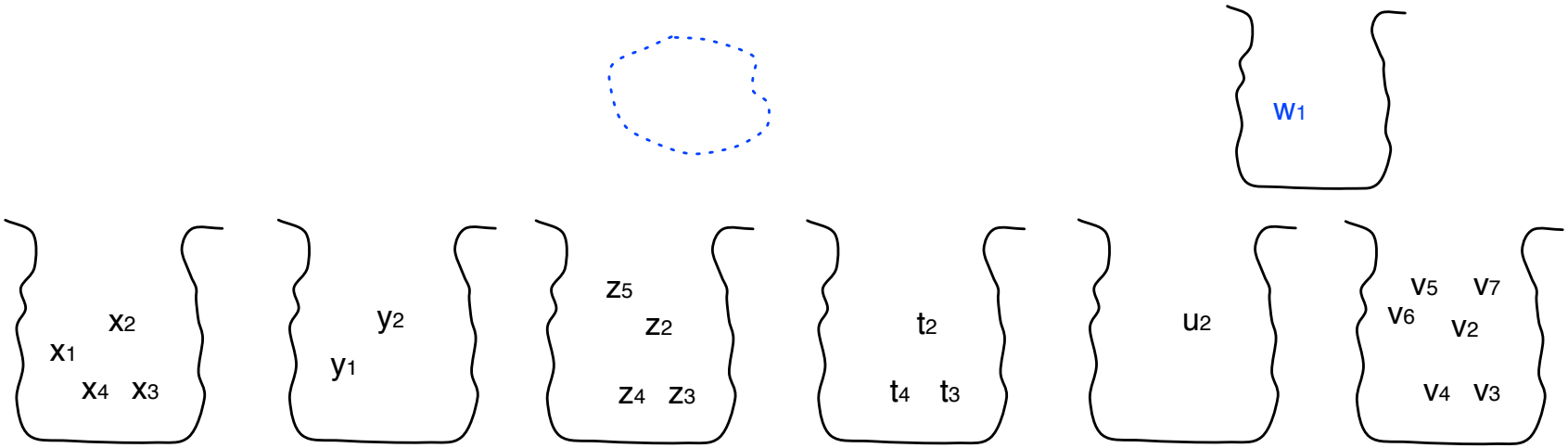
Finding reactants (with a simple condition)

replace x,y,z,t,u,v **by** f(x,y,z) **if** c_1(x,y)



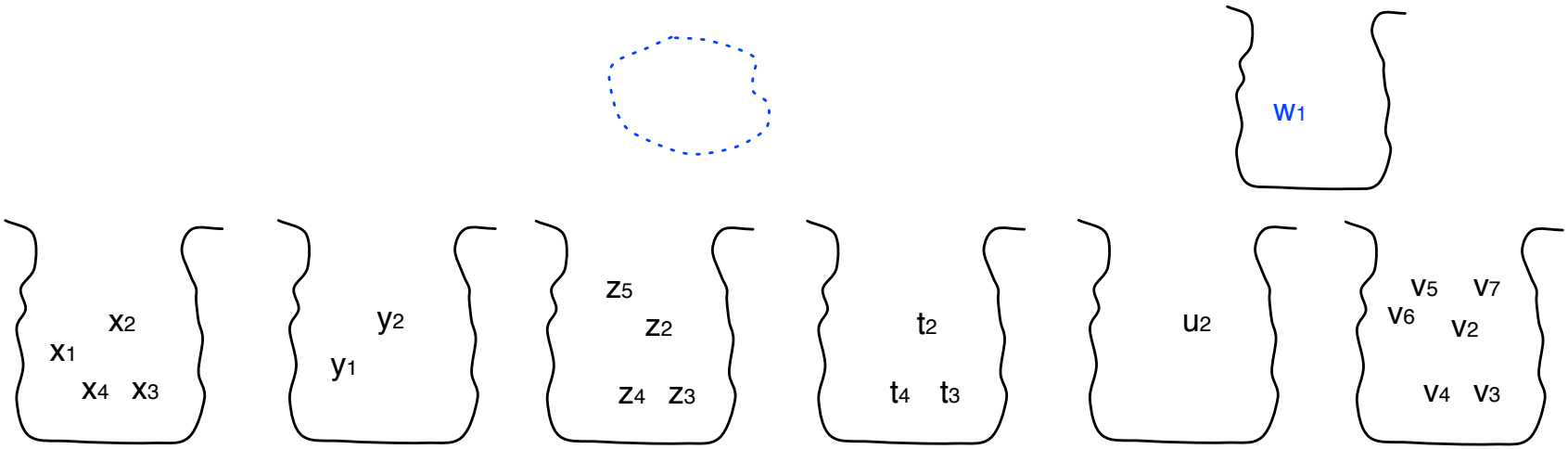
Finding reactants (with a simple condition)

replace x,y,z,t,u,v **by** f(x,y,z) **if** c_1(x,y)



Finding reactants (with a simple condition)

replace x,y,z,t,u,v **by** f(x,y,z) **if** c_1(x,y)



When there is 1 condition on 2 molecules, detecting inertia is quadratic.

Finding reactants (with a composite condition)

```
replace  x,y,z,t,u,v  by  f(x,y,z)  if  [ c1(x,y)  and
                                          c2(y,z)  and
                                          c3(z,t)  and
                                          c4(z,u)  and
                                          c5(t,v) ]
                                    or c6(x,t,u)
```

- Any condition can be put into disjunctive normal form
- Each conjunctive clause can be explored independently
- The constraints can be displayed as a graph

Finding reactants (with a composite condition)

```
replace  x,y,z,t,u,v  by  f(x,y,z)  if  c1(x,y)  and
                                        c2(y,z)  and
                                        c3(z,t)  and
                                        c4(z,u)  and
                                        c5(t,v)

replace  x,y,z,t,u,v  by  f(x,y,z)  if  c6(x,t,u)
```

- Any condition can be put into disjunctive normal form
- Each conjunctive clause can be explored independently
- The constraints can be displayed as a graph

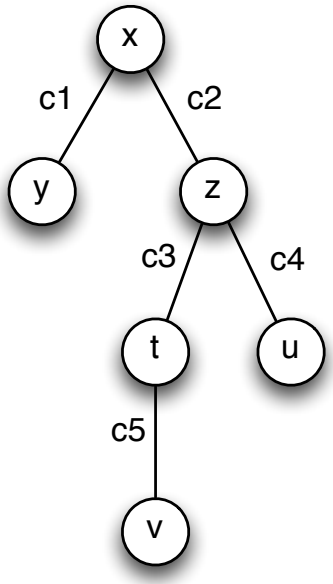
Finding reactants (with a composite condition)

```
replace x,y,z,t,u,v by f(x,y,z) if c1(x,y) and
                                   c2(y,z) and
                                   c3(z,t) and
                                   c4(z,u) and
                                   c5(t,v)
```

- Any condition can be put into disjunctive normal form
- Each conjunctive clause can be explored independently
- The constraints can be displayed as a graph

Finding reactants (with a composite condition)

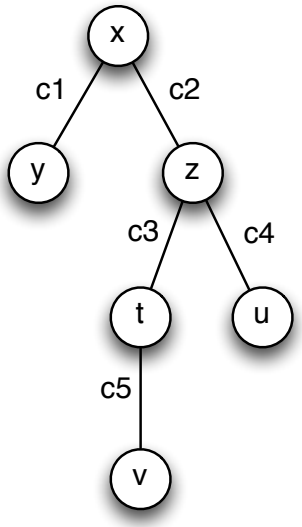
```
replace x,y,z,t,u,v by f(x,y,z) if c1(x,y) and
                                   c2(y,z) and
                                   c3(z,t) and
                                   c4(z,u) and
                                   c5(t,v)
```



- Any condition can be put into disjunctive normal form
- Each conjunctive clause can be explored independently
- The constraints can be displayed as a graph

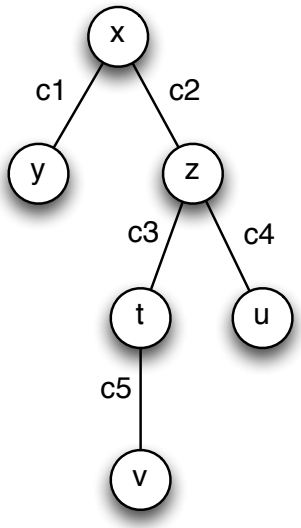
Finding reactants is a matter of graph comparison

The rule

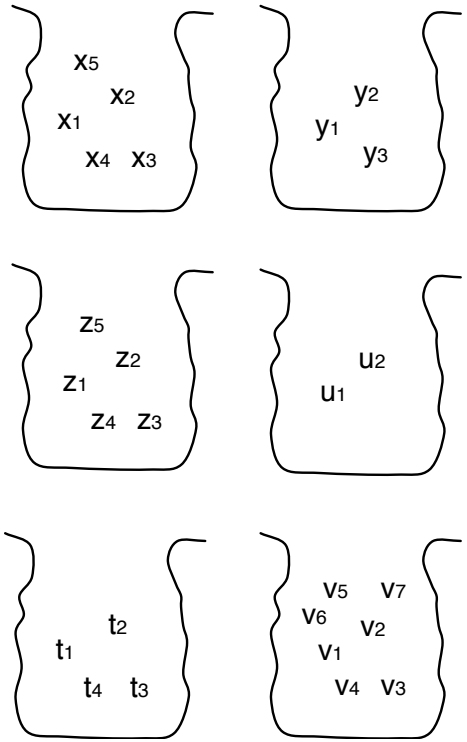


Finding reactants is a matter of graph comparison

The rule

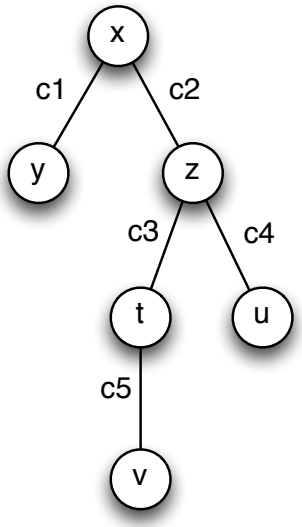


The molecules

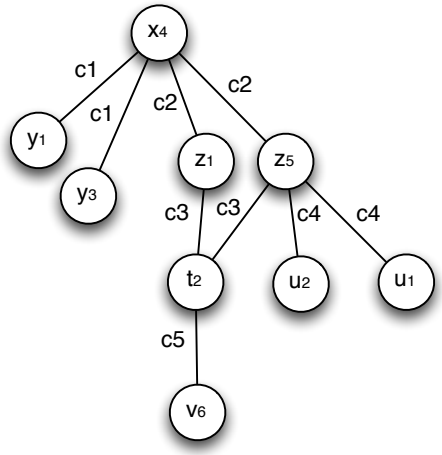


Finding reactants is a matter of graph comparison

The rule



Building the graph of molecules

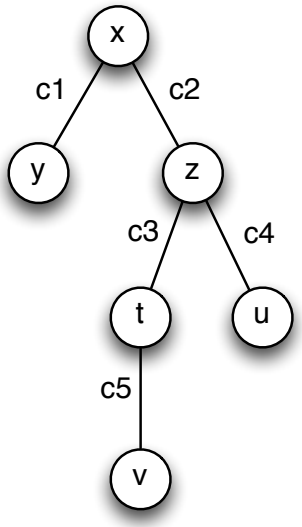


Reactants Searching Problem $\equiv$ *Subgraph Isomorphism Problem*

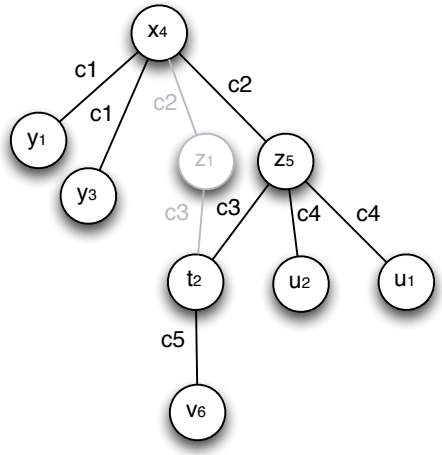
(We need to find the rule graph inside the molecules graph)

Finding reactants is a matter of graph comparison

The rule



Cleaning the graph

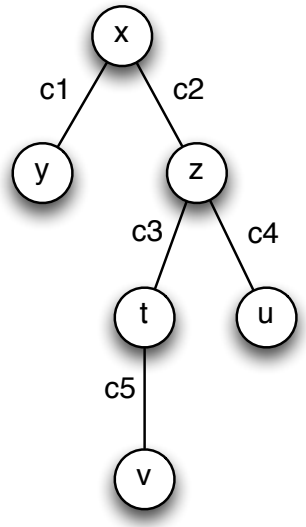


Reactants Searching Problem $\equiv$ *Subgraph Isomorphism Problem*

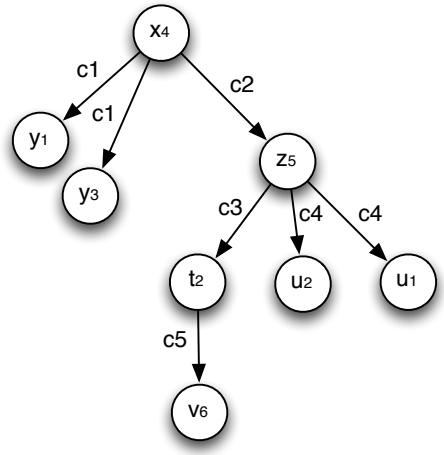
(We need to find the rule graph inside the molecules graph)

Finding reactants is a matter of graph comparison

The rule



Traversing the graph

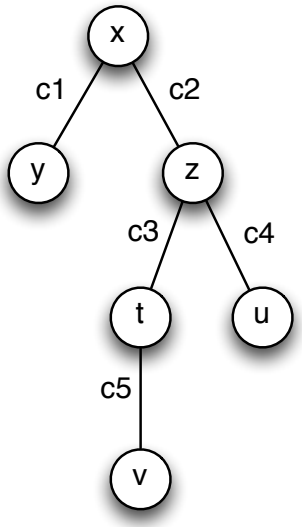


Reactants Searching Problem $\equiv$ *Subgraph Isomorphism Problem*

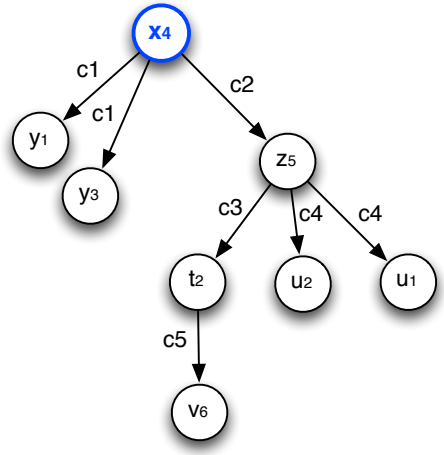
(We need to find the rule graph inside the molecules graph)

Finding reactants is a matter of graph comparison

The rule



Traversing the graph

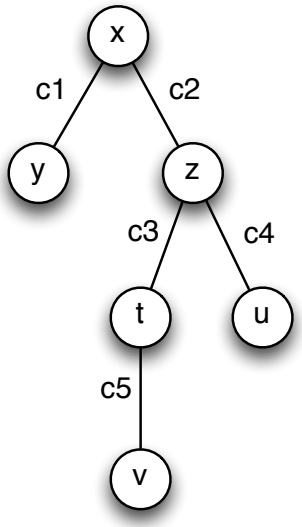


Reactants Searching Problem $\equiv$ *Subgraph Isomorphism Problem*

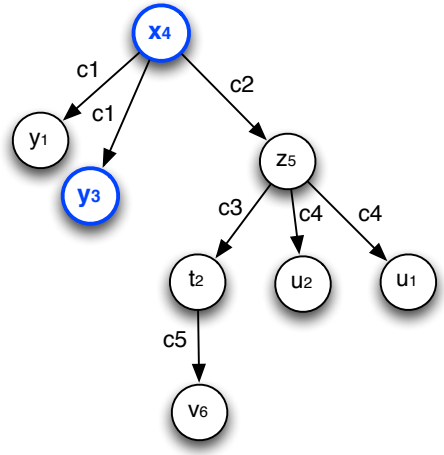
(We need to find the rule graph inside the molecules graph)

Finding reactants is a matter of graph comparison

The rule



Traversing the graph

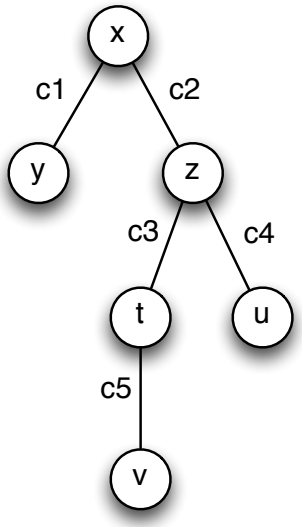


Reactants Searching Problem ≡ *Subgraph Isomorphism Problem*

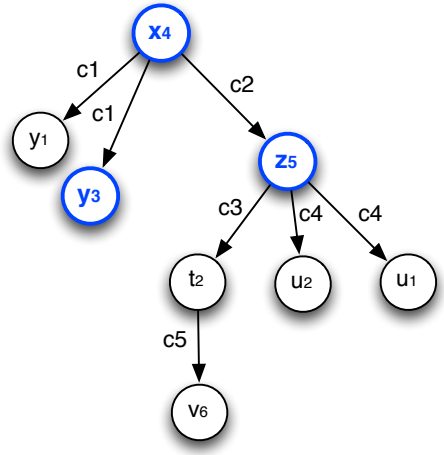
(We need to find the rule graph inside the molecules graph)

Finding reactants is a matter of graph comparison

The rule



Traversing the graph

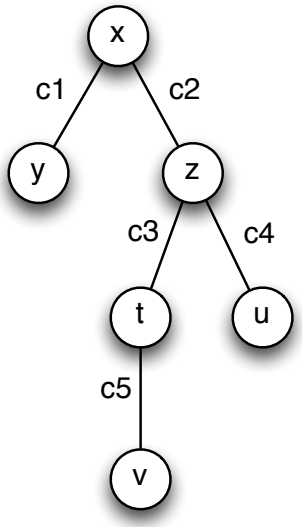


Reactants Searching Problem ≡ *Subgraph Isomorphism Problem*

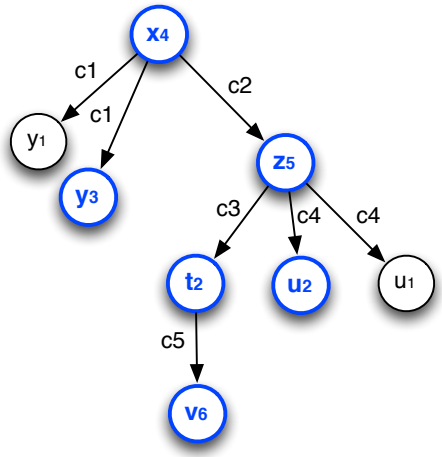
(We need to find the rule graph inside the molecules graph)

Finding reactants is a matter of graph comparison

The rule



Reactants found

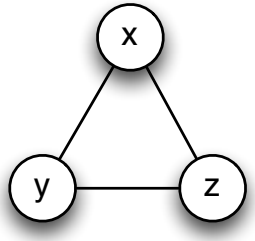


Reactants Searching Problem $\equiv$ *Subgraph Isomorphism Problem*

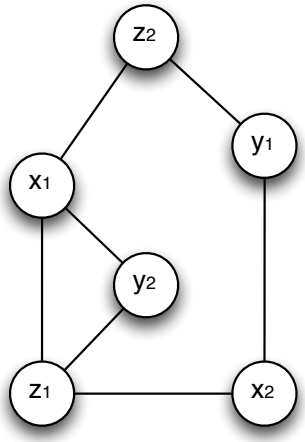
(We need to find the rule graph inside the molecules graph)

Towards more complicated rules

The rule

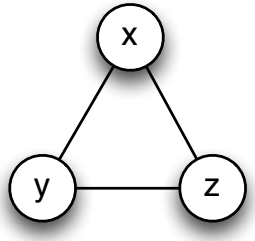


The molecules

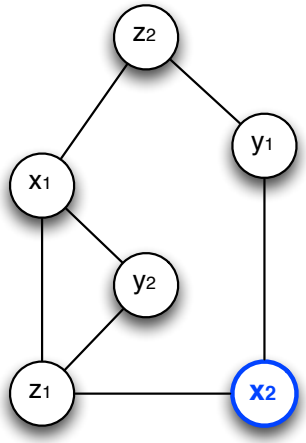


Towards more complicated rules

The rule

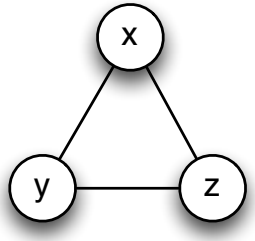


The molecules

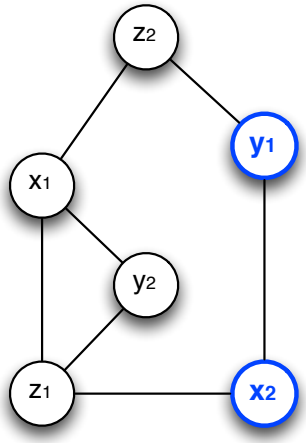


Towards more complicated rules

The rule

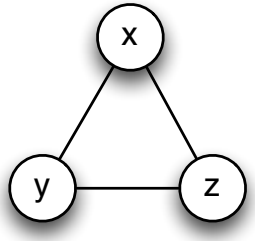


The molecules

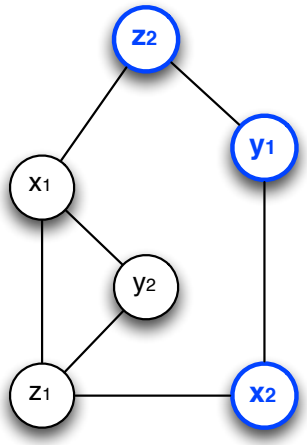


Towards more complicated rules

The rule

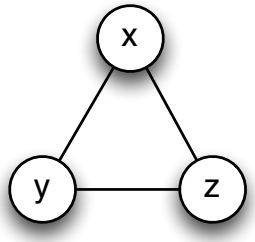


The molecules

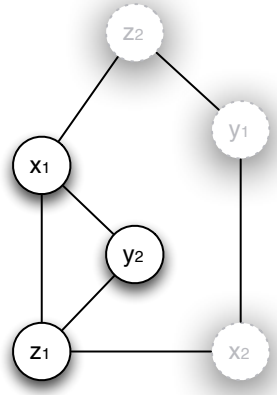


Towards more complicated rules

The rule

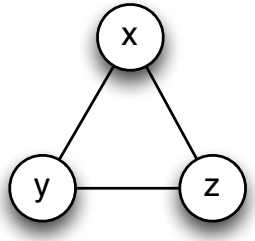


The molecules

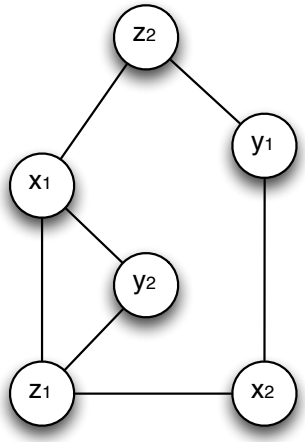


Towards more complicated rules

The rule

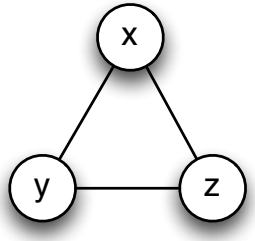


The molecules

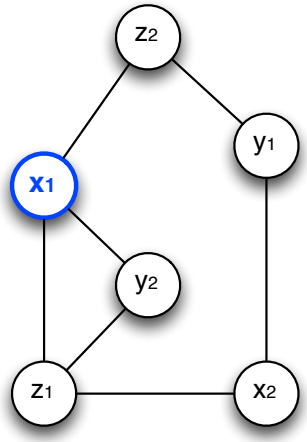


Towards more complicated rules

The rule

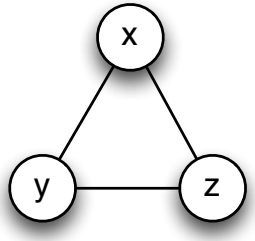


The molecules

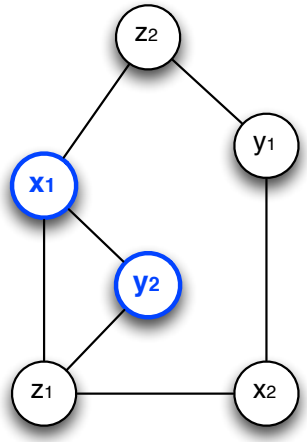


Towards more complicated rules

The rule

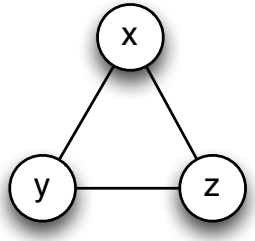


The molecules

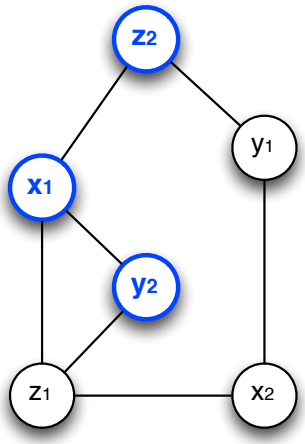


Towards more complicated rules

The rule

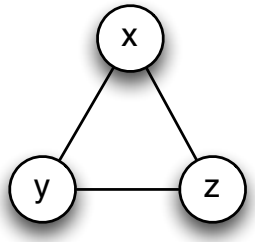


The molecules

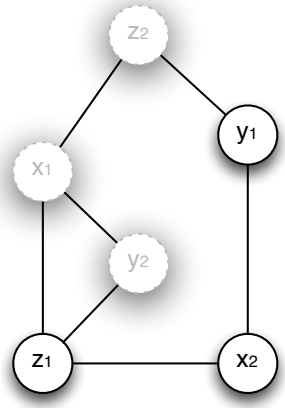


Towards more complicated rules

The rule

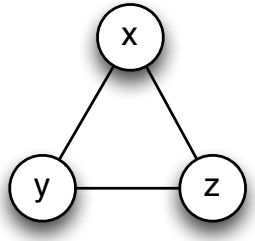


The molecules

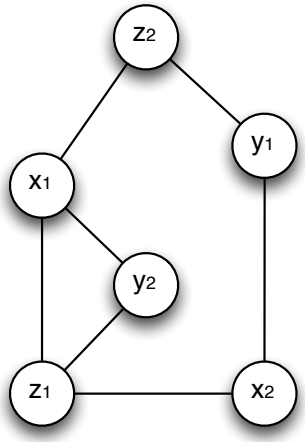


Towards more complicated rules

The rule

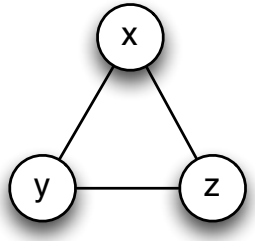


The molecules

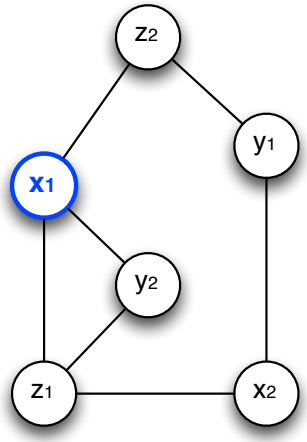


Towards more complicated rules

The rule

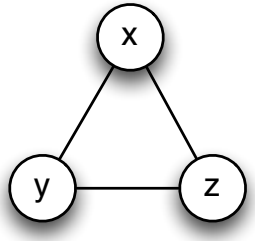


The molecules

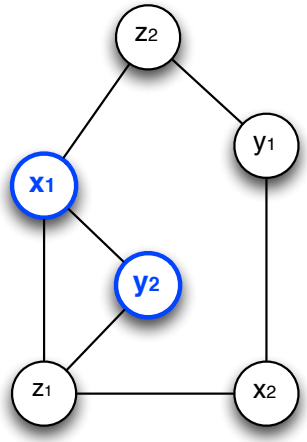


Towards more complicated rules

The rule

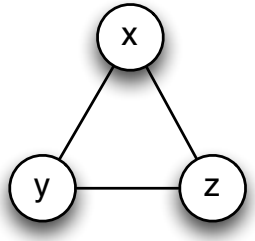


The molecules

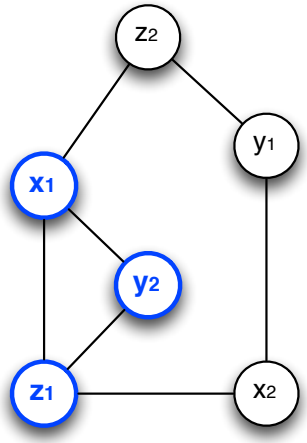


Towards more complicated rules

The rule

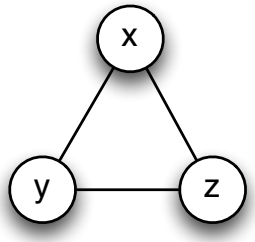


Reactants found!

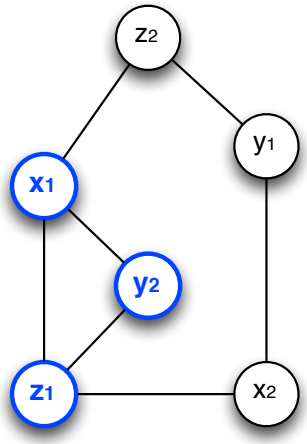


Towards more complicated rules

The rule

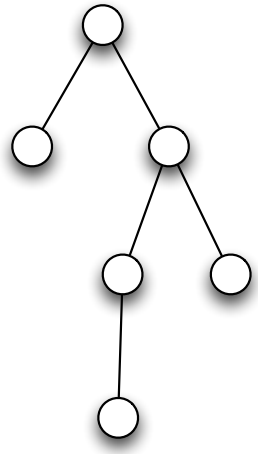


Reactants found!



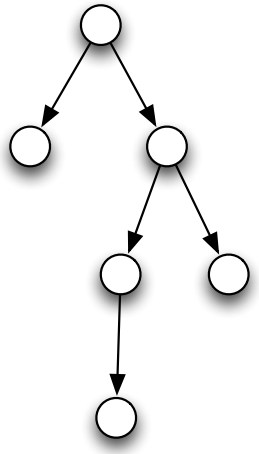
Greedy selection does not seem to work anymore.
Are we falling back to the basic brute force search?

Different graphs, different complexities



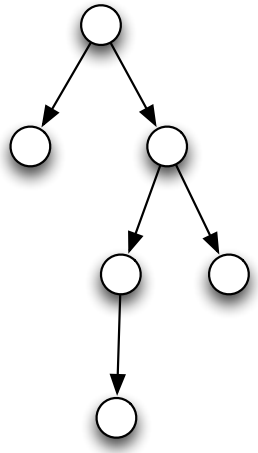
- Trees are easy
- Cycles may invalidate choices
  - Orienting the graph
  - 2-HEAVY nodes appear
  - molecules whose initial choice may be invalidated
  - Goal: minimize the number C of such nodes

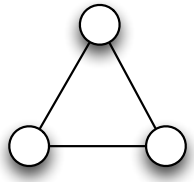
Different graphs, different complexities



- Trees are easy
- Cycles may invalidate choices
  - Orienting the graph
  - 2-HEAVY nodes appear
  - molecules whose initial choice may be invalidated
  - Goal: minimize the number C of such nodes

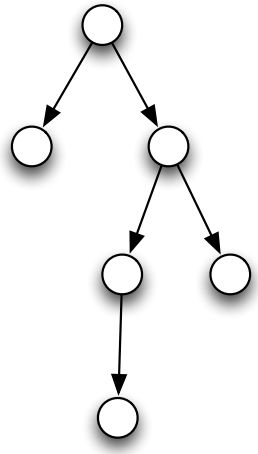
Different graphs, different complexities

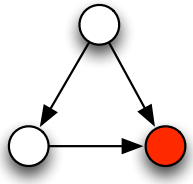




- Trees are easy
- Cycles may invalidate choices
  - Orienting the graph
  - 2-HEAVY nodes appear
  - molecules whose initial choice may be invalidated
  - Goal: minimize the number C of such nodes

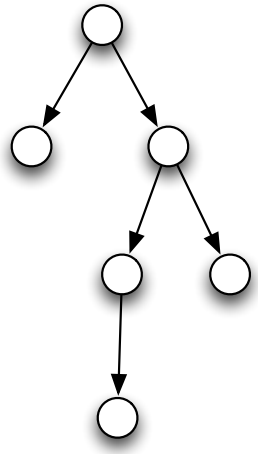
Different graphs, different complexities

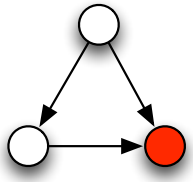


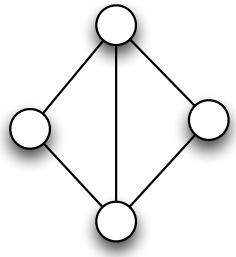


- Trees are easy
- Cycles may invalidate choices
  - Orienting the graph
  - 2-HEAVY nodes appear
  - molecules whose initial choice may be invalidated
  - Goal: minimize the number C of such nodes

Different graphs, different complexities

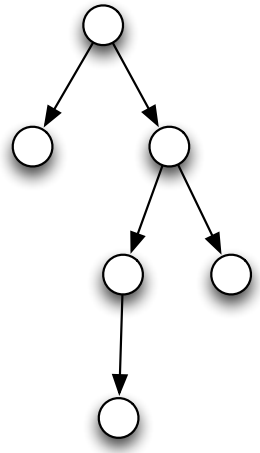


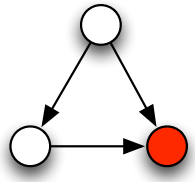


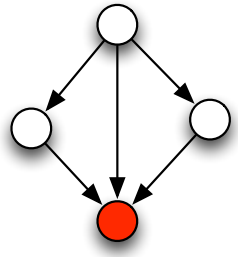


- Trees are easy
- Cycles may invalidate choices
  - Orienting the graph
  - 2-HEAVY nodes appear
  - molecules whose initial choice may be invalidated
  - Goal: minimize the number C of such nodes

Different graphs, different complexities





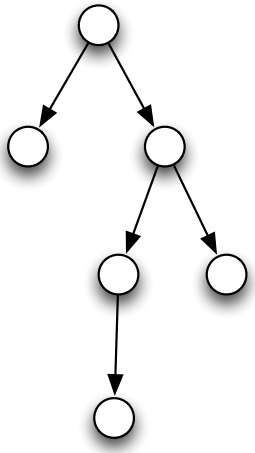
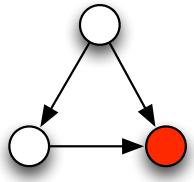
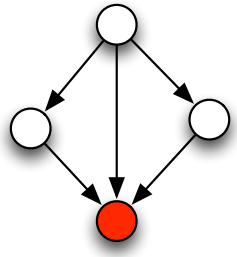
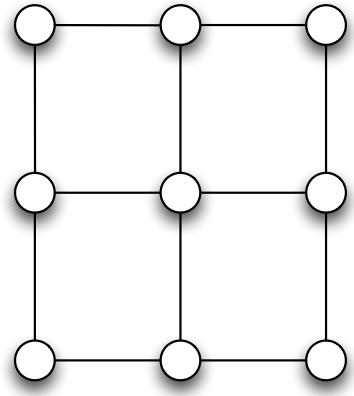


- Trees are easy
- Cycles may invalidate choices
  - Orienting the graph
  - 2-HEAVY nodes appear
  - molecules whose initial choice may be invalidated
  - Goal: minimize the number $\mathtt{C}$ of such nodes

Theorem

$\mathtt{C} \leq k - 2$

Different graphs, different complexities

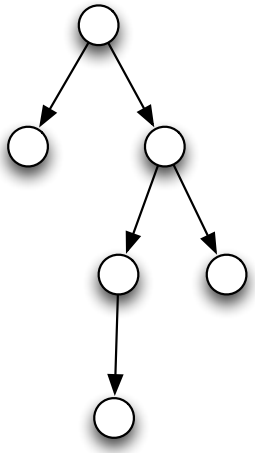
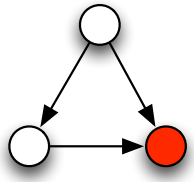
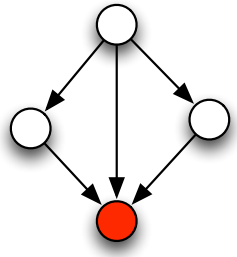
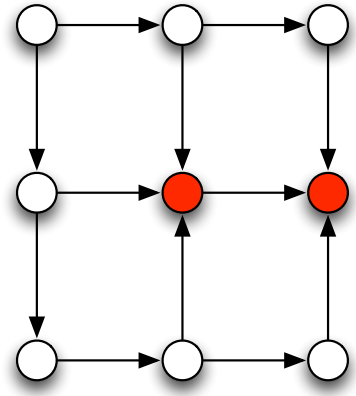





- Trees are easy
- Cycles may invalidate choices
  - Orienting the graph
  - 2-HEAVY nodes appear
  - molecules whose initial choice may be invalidated
  - Goal: minimize the number $\mathtt{C}$ of such nodes

Theorem

$\mathtt{C} \leq k - 2$

Different graphs, different complexities

- Trees are easy
- Cycles may invalidate choices
  - Orienting the graph
  - 2-HEAVY nodes appear
  - molecules whose initial choice may be invalidated
  - Goal: minimize the number $\mathtt{C}$ of such nodes

Theorem

$\mathtt{C} \leq k - 2$

An algorithm for RSP

Algorithm

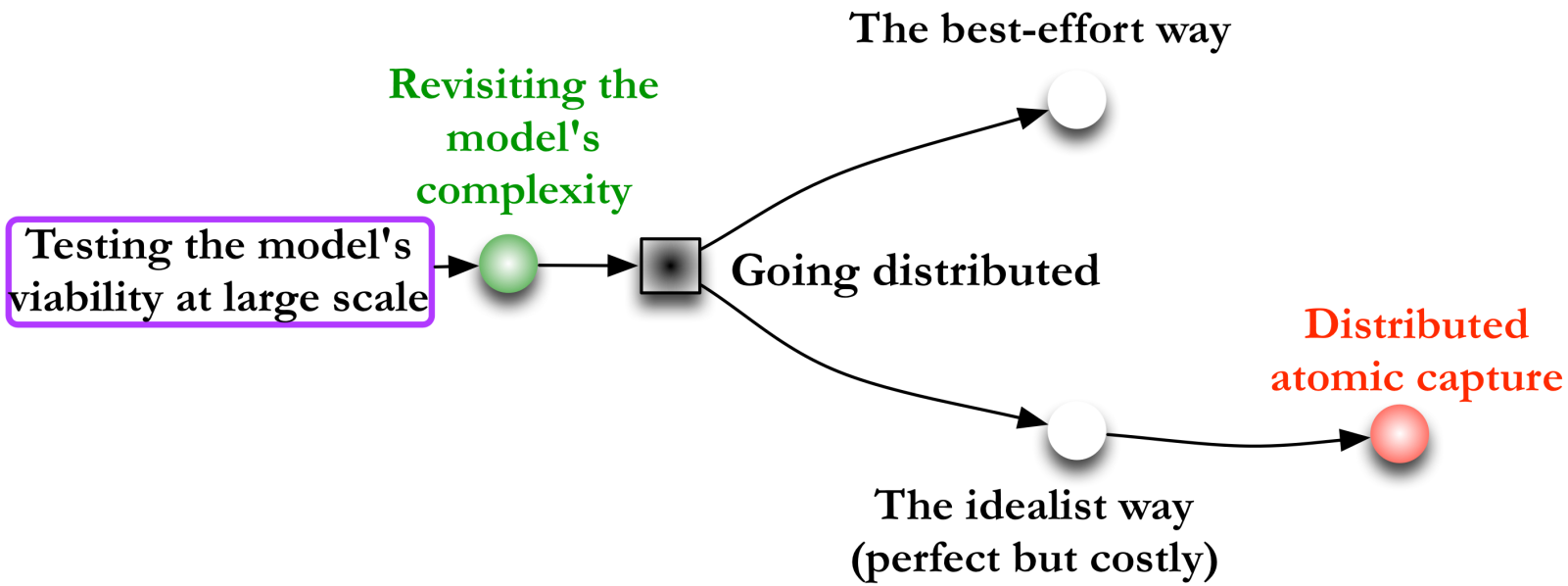
For each possible assignment of 2-HEAVY nodes

1. clean the graph
2. if the remaining graph is reactive, choose the other molecules greedily

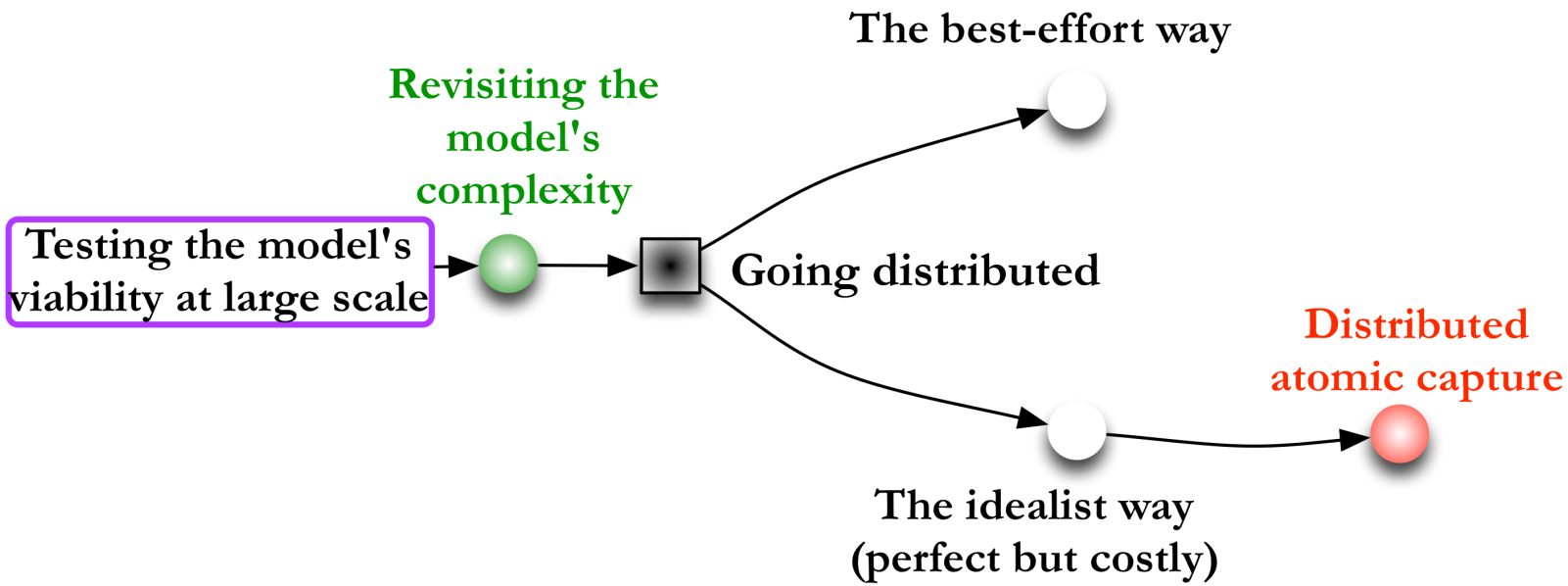
Complexity

- $O(n^{c+2})$
- $c + 2 = k$ for K_k
- $c + 2 < k$ otherwise (*most* of the time)

Part II


The best-effort way
Revisiting the model's complexity
Testing the model's viability at large scale
Going distributed
Distributed atomic capture
The idealist way (perfect but costly)

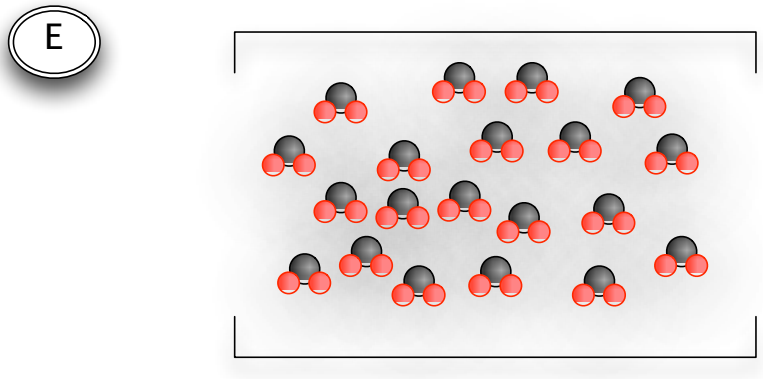
Part II



Joint work with
- Marko Obrovac (PhD, U. Rennes 1)
- Marin Bertier (INSA Rennes)

Towards physical distribution

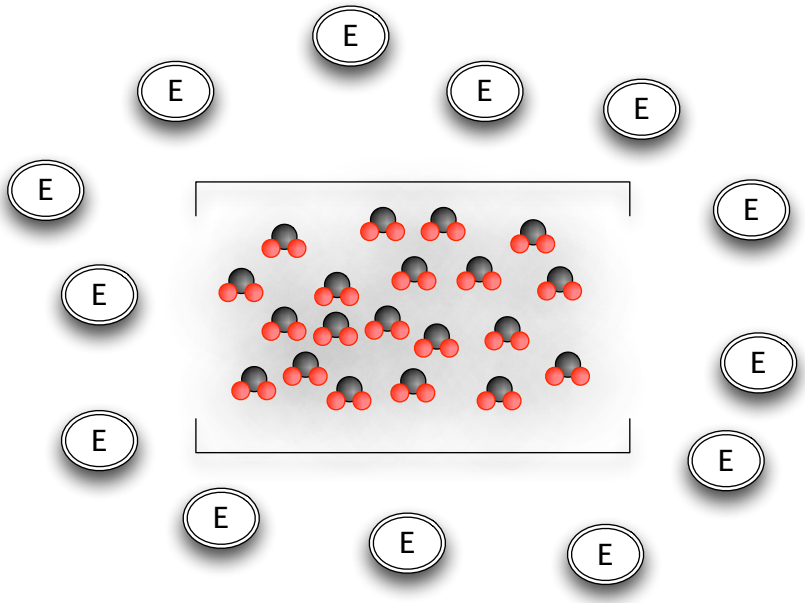
Acquiring molecules?



- Single threaded
- Multi-threaded / shared memory
- Distributed memory?

Towards physical distribution

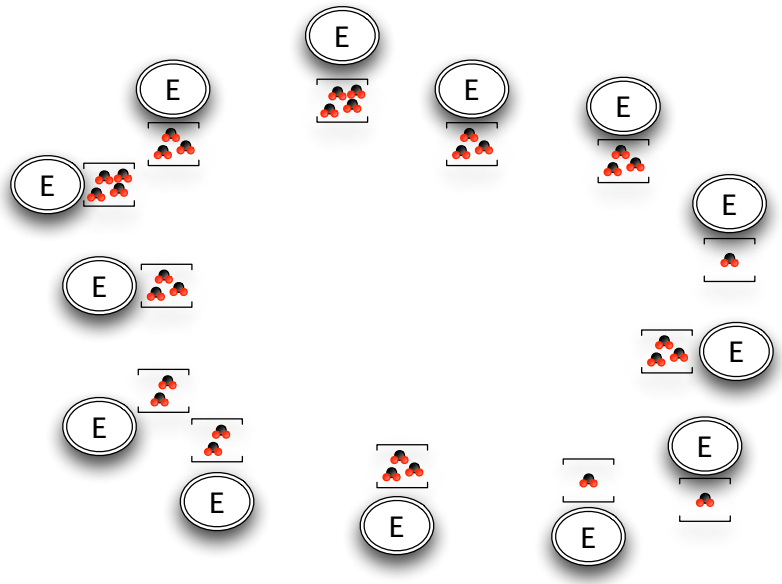
Acquiring molecules?



- Single threaded
- Multi-threaded / shared memory
- Distributed memory?

Towards physical distribution

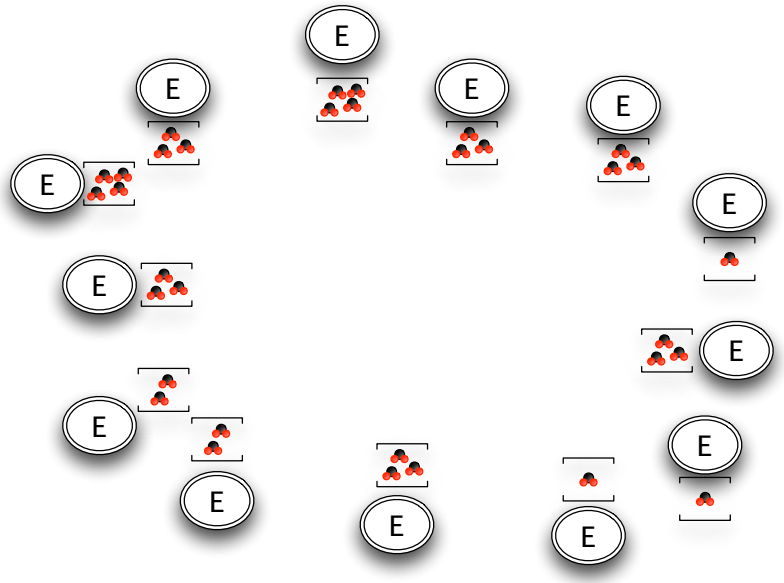
Acquiring molecules?



- Single threaded
- Multi-threaded / shared memory
- Distributed memory?

Towards physical distribution

Acquiring molecules?



- Single threaded
- Multi-threaded / shared memory
- Distributed memory?

The phases

1. Finding matching molecules (through the *discovery protocol*)
2. Obtaining them (through the *capture protocol*)

The problem

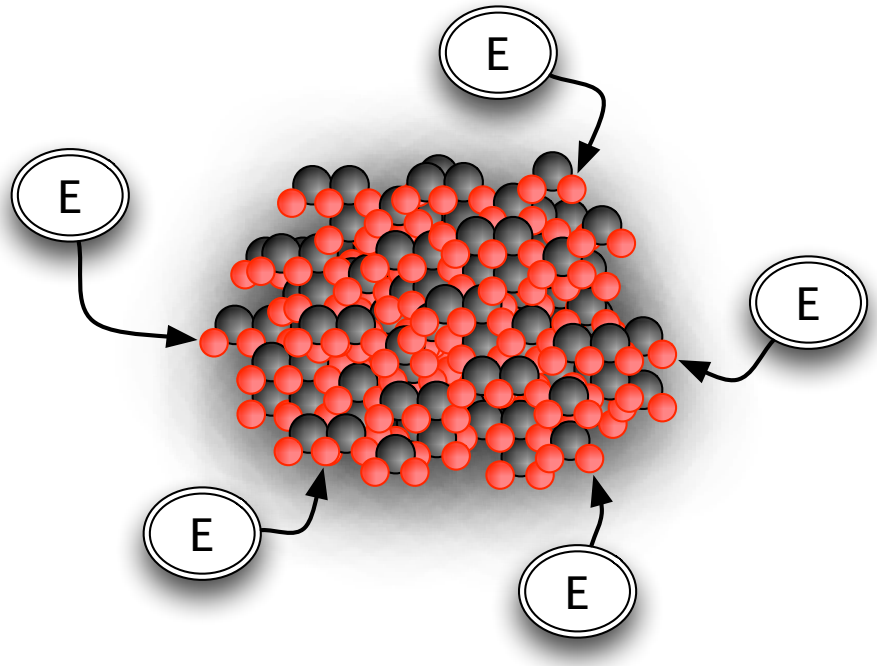
Looks like the *drinking philosophers problem* [Chandy & Misra (1984)]

- philosophers $\equiv$ engines
- bottles $\equiv$ molecules
- cocktails $\equiv$ reactants

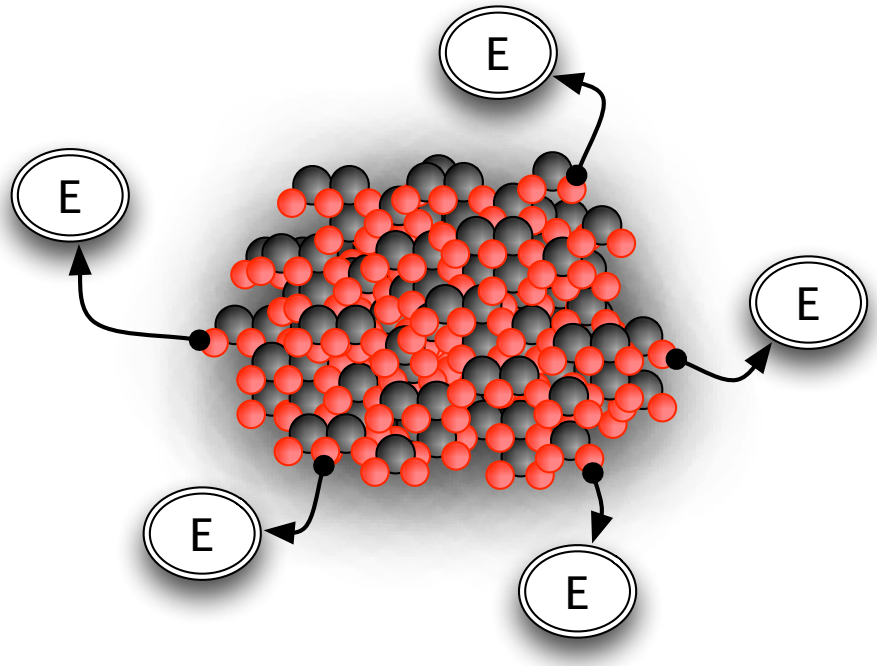
But ...

- Molecules are interchangeable to some extent
  (also close to the *K-out-of-M* problem [Raynal (1991)])
- Molecules are dynamic
- We need liveness but not starvation-freeness
- The concentration of molecules varies

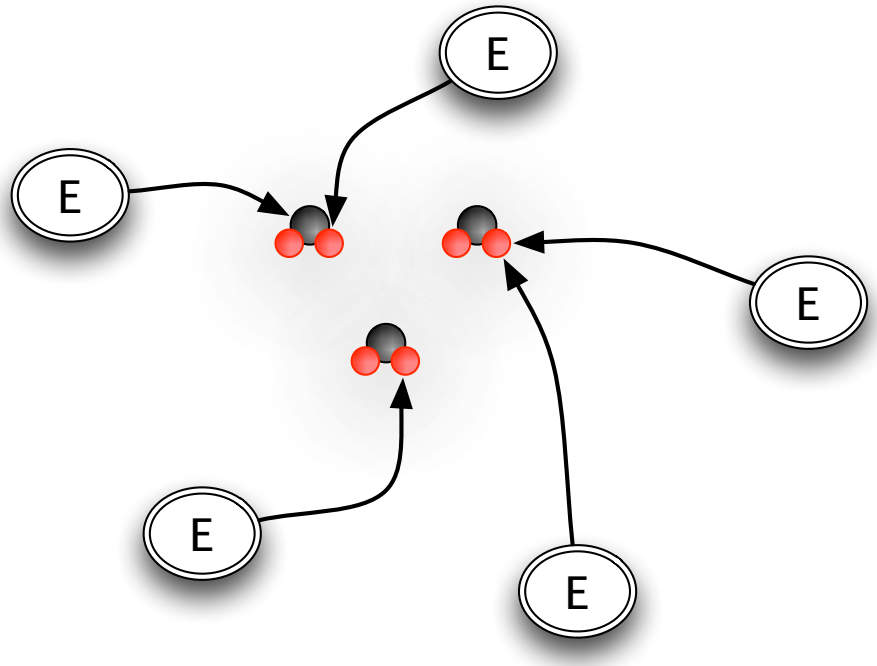
Variability of molecules' concentration



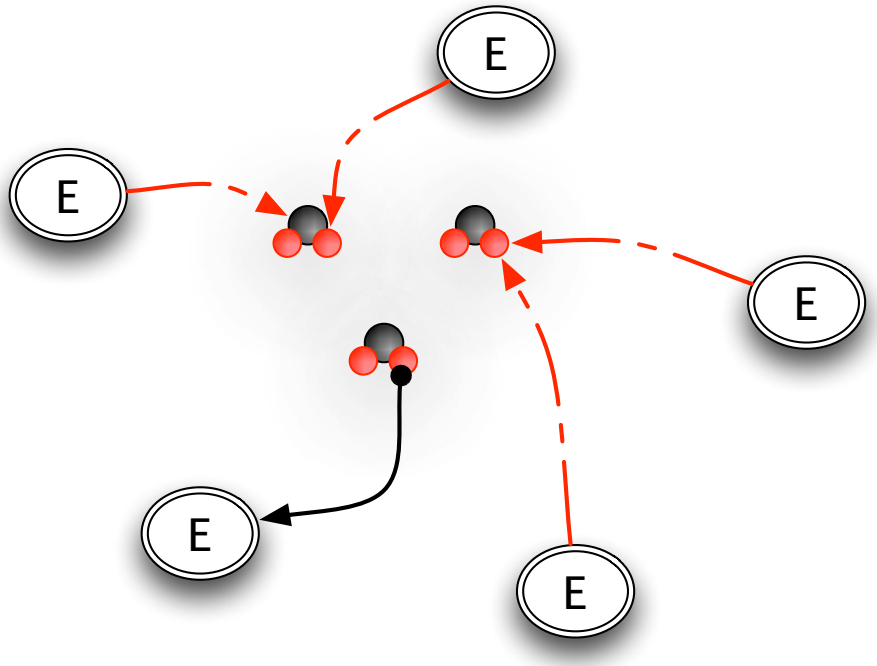
Variability of molecules' concentration



Variability of molecules' concentration



Variability of molecules' concentration



The capture protocol

We need a **less restrictive** but **adaptive** solution.

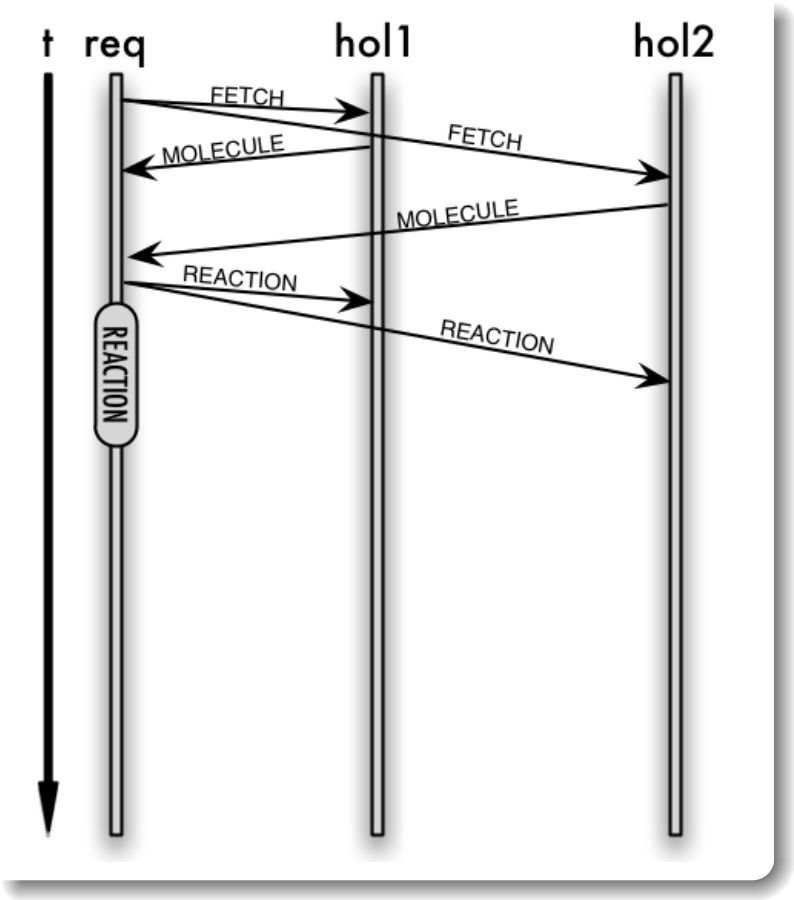
Entities

- Molecule's holder
- Molecule's requester
- FIFO reliable channels

Two protocols

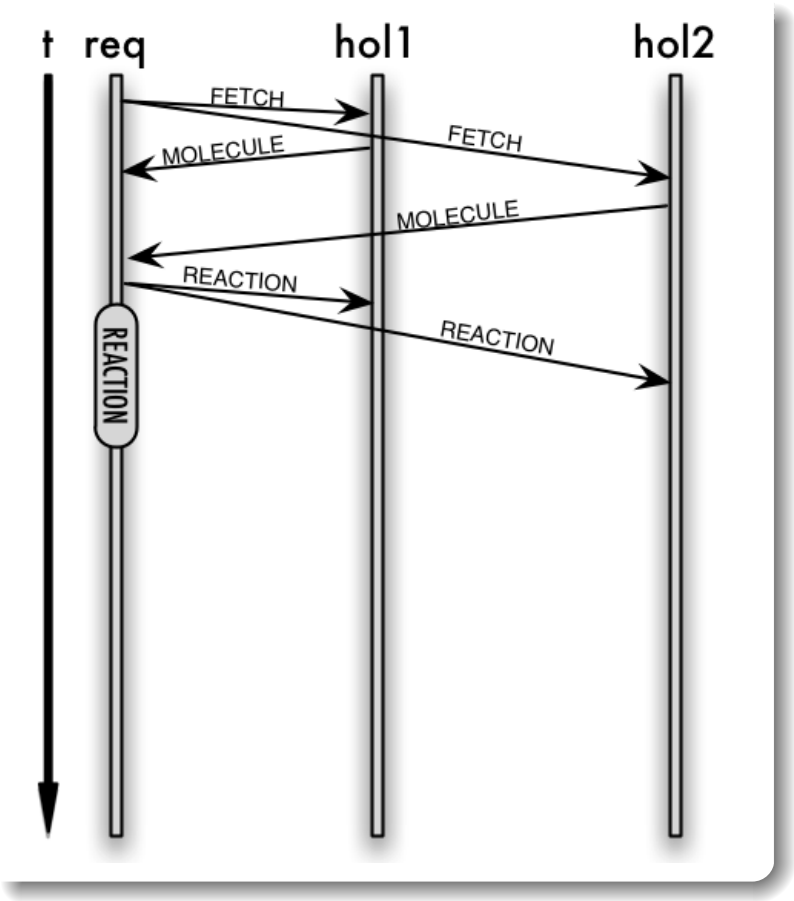
- An optimistic protocol to progress faster
- A pessimistic protocol to ensure liveness
- Decentralized and dynamic switching between protocols

The optimistic protocol

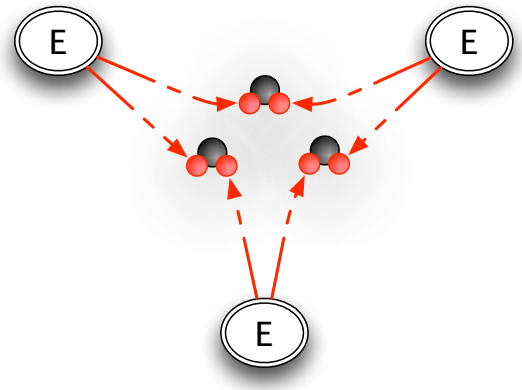


- For high reactants concentration
- For low conflict rates
- Simple and fast
- Solve conflicts for one molecule (FCFS)
- Does not ensure liveness

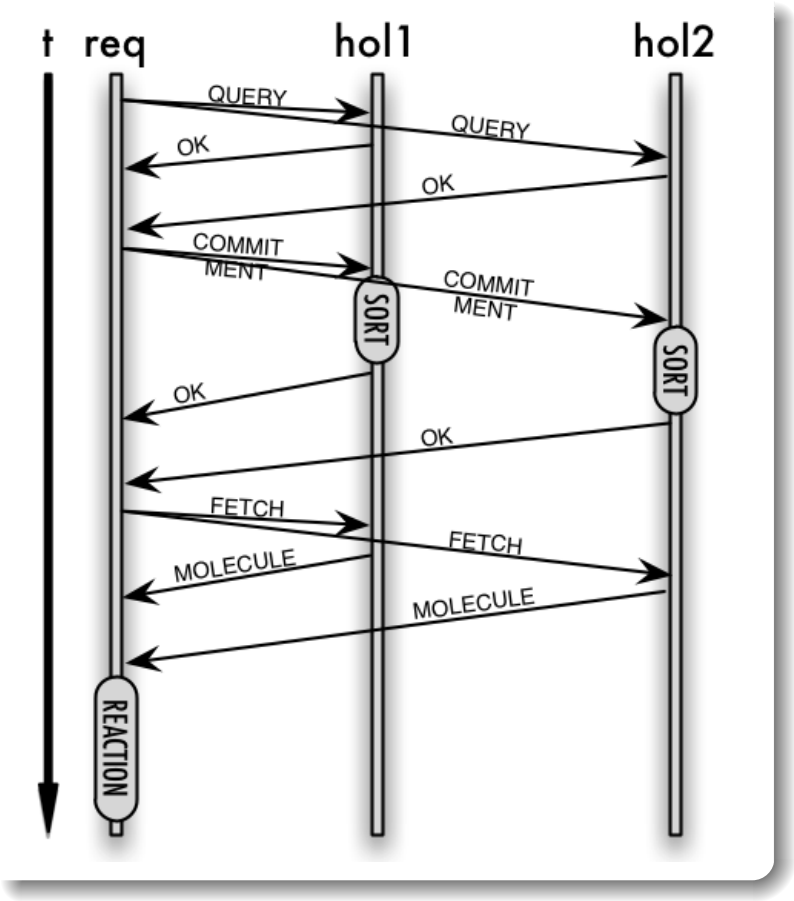
The optimistic protocol



- For high reactants concentration
- For low conflict rates
- Simple and fast
- Solve conflicts for one molecule (FCFS)
- Does not ensure liveness



The pessimistic protocol



- For low reactants concentration
- For high conflict rates
- Ensures liveness
- A three-phase decision protocol
- At least one node gets its reaction done

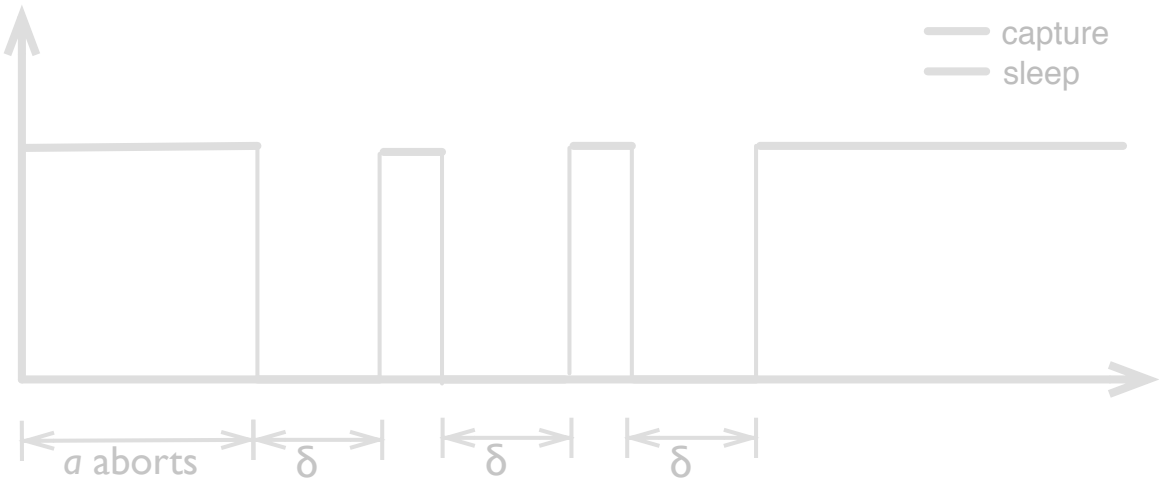
Adaptation to concentration's variability

Switching between protocols is a local decision based on observed success rate, given σ the observed success rate, k the rule's arity, a threshold k

optimist $\xrightarrow{\sigma^k \leq s}$ pessimist

pessimist $\xrightarrow{\sigma^k \geq s}$ optimist

After a certain amount of aborts, nodes become dormant for a little while



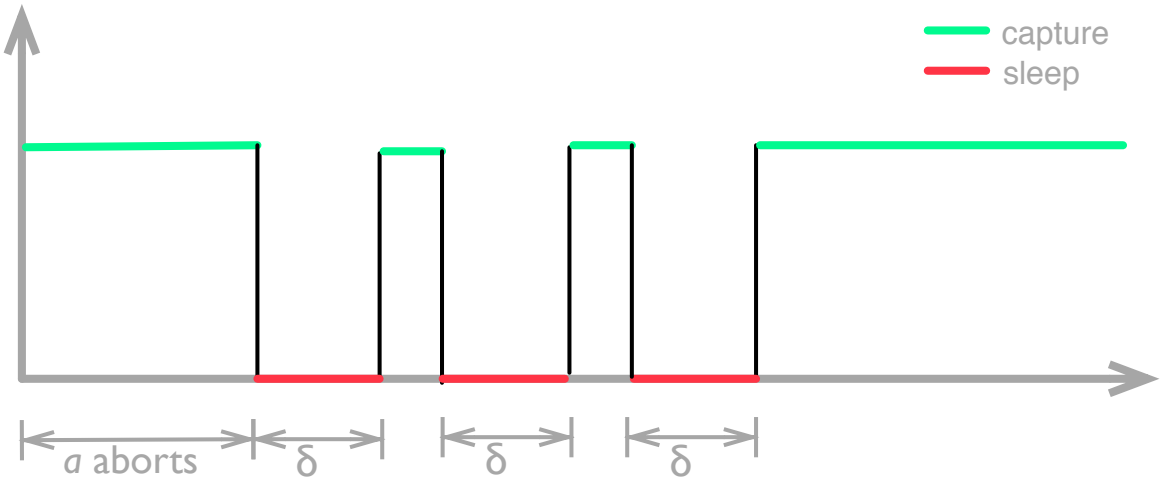
Adaptation to concentration's variability

Switching between protocols is a local decision based on observed success rate, given σ the observed success rate, k the rule's arity, a threshold k

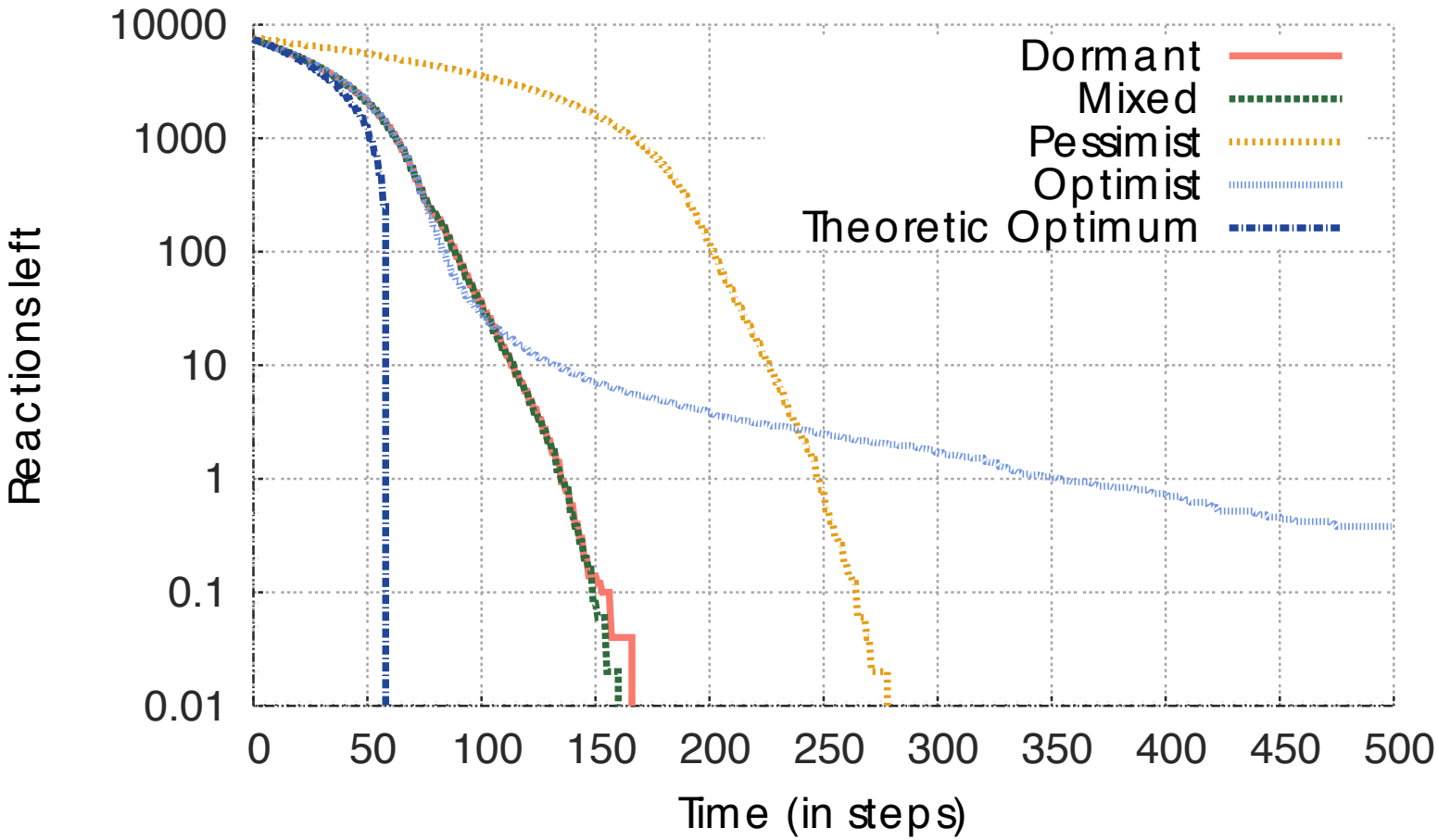
optimist $\xrightarrow{\sigma^k \leq s}$ pessimist

pessimist $\xrightarrow{\sigma^k \geq s}$ optimist

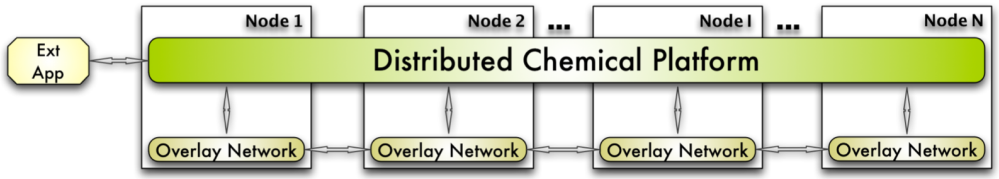
After a certain amount of aborts, nodes become dormant for a little while



Simulation results (overall performance)



Design of a distributed chemical machine

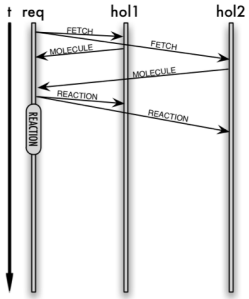


PhD Marko Obrovac

Going distributed

The idealist way
(perfect but costly)

Distributed atomic capture

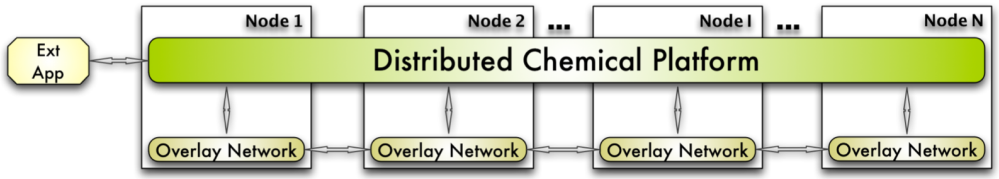


Distributed discovery
and inertia detection

Higher-order

Software
protoype

Design of a distributed chemical machine

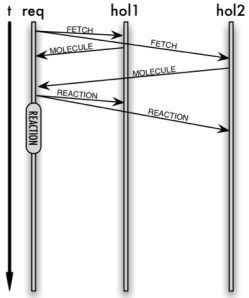


PhD Marko Obrovac

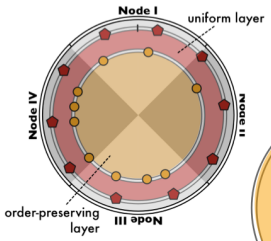
Going distributed

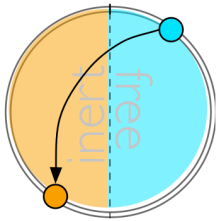
The idealist way (perfect but costly)

Distributed atomic capture



Distributed discovery and inertia detection

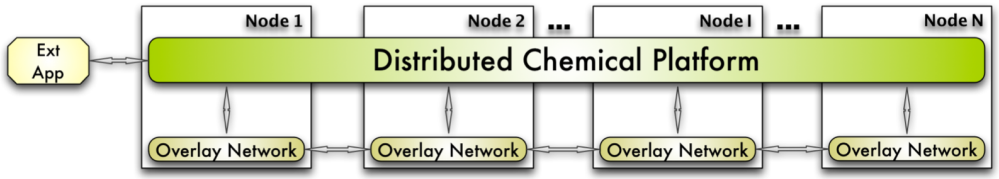




Higher-order

Software prototype

Design of a distributed chemical machine

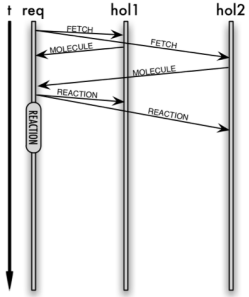


PhD Marko Obrovac

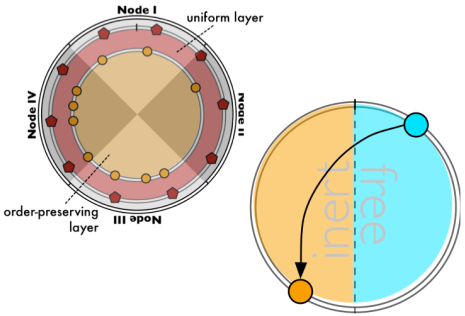
Going distributed

The idealist way (perfect but costly)

Distributed atomic capture



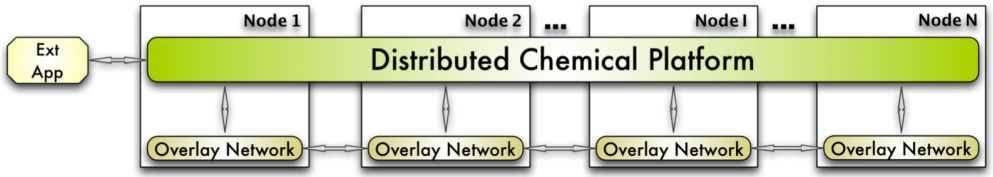
Distributed discovery and inertia detection



Higher-order

Software protoype

Design of a distributed chemical machine

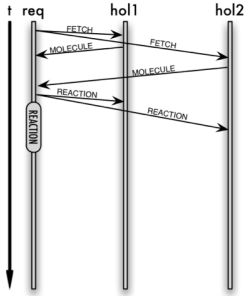


PhD Marko Obrovac

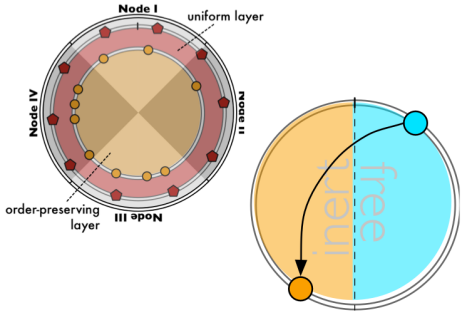
Going distributed

The idealist way
(perfect but costly)

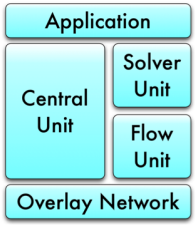
Distributed atomic capture



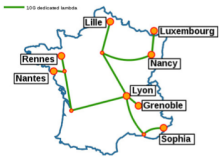
Distributed discovery and inertia detection



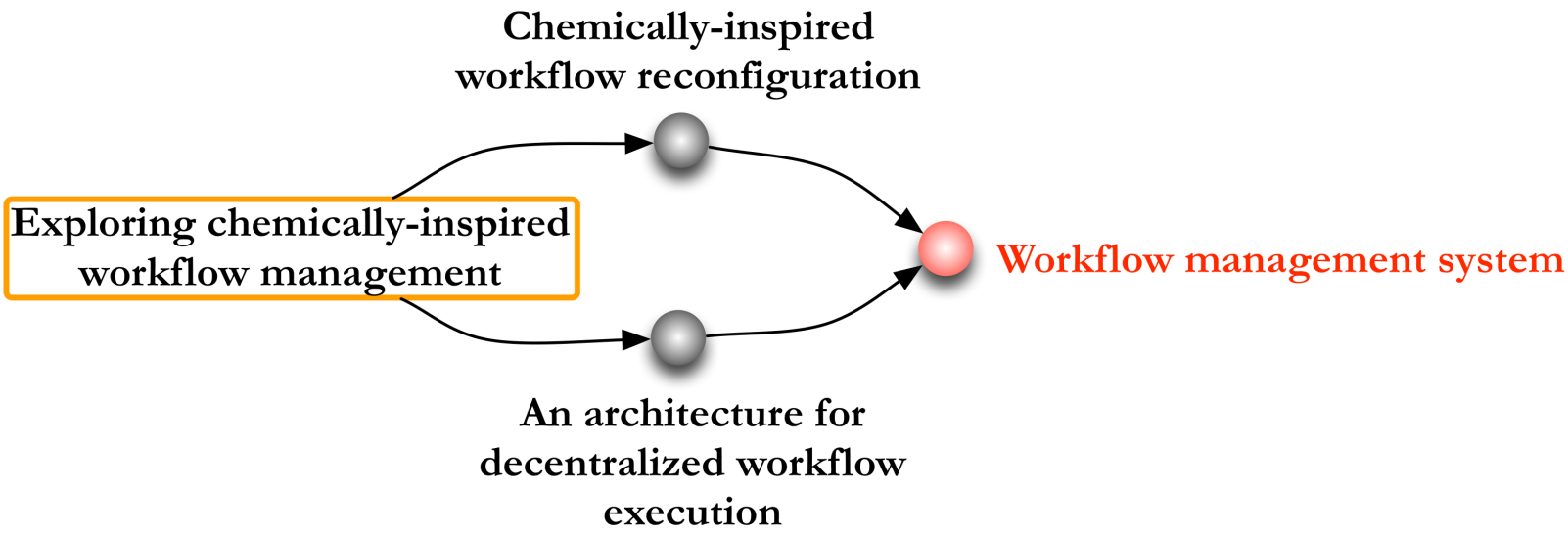
Higher-order



Software protoype



Part III


Chemically-inspired
workflow reconfiguration
Exploring chemically-inspired
workflow management
Workflow management system
An architecture for
decentralized workflow
execution

Part III

Post-doc Javier Rojas Balderrama

Chemically-inspired workflow reconfiguration

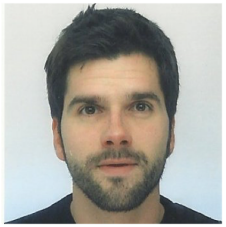
Exploring chemically-inspired workflow management

Workflow management system

GINFLOW

Inria ADT Matthieu Simonin

An architecture for decentralized workflow execution

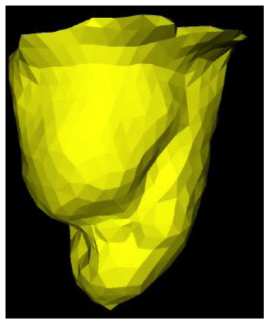
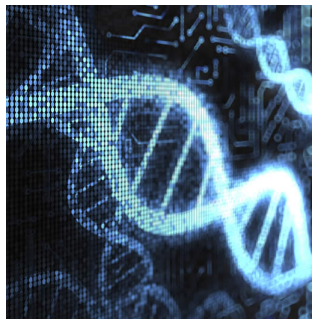


PhD Héctor Fernandez

Scientific workflows

Workflows are common in computer-assisted scientific experiments.

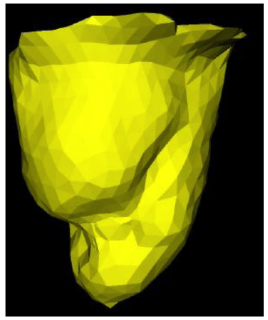
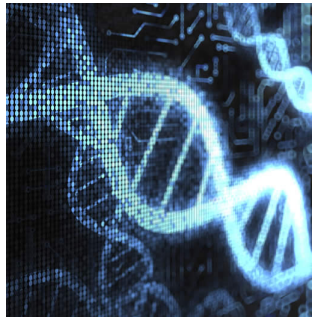
- astronomy
- medical imaging
- bioinformatics
- ...

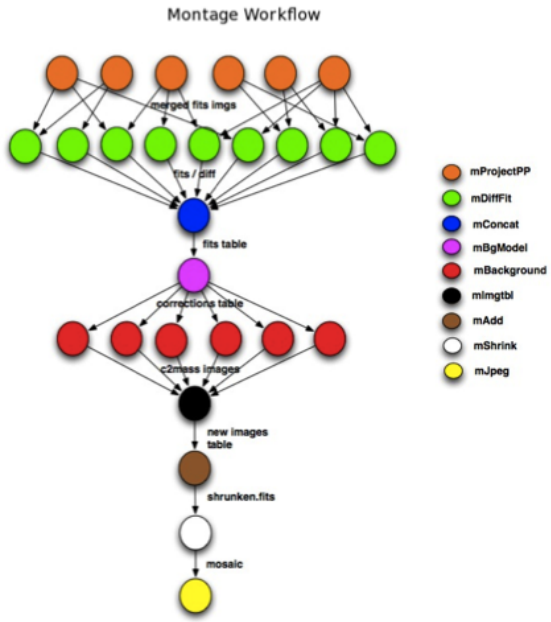




Scientific workflows

Workflows are common in computer-assisted scientific experiments.

- astronomy
- medical imaging
- bioinformatics
- ...

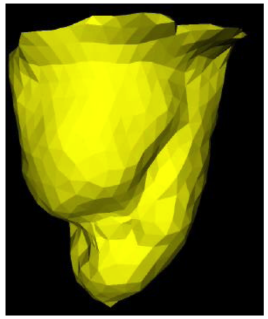
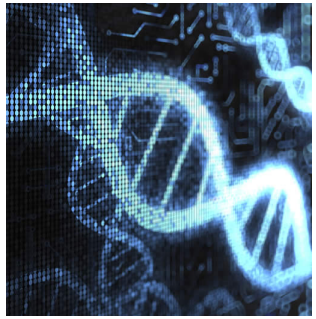






Scientific workflows

Workflows are common in computer-assisted scientific experiments.

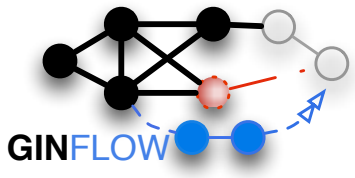
- astronomy
- medical imaging
- bioinformatics
- ...

Some workflow management systems are now mature and broadly used.


- Galaxy
- Kepler
- Moteur
- Pegasus
- Taverna

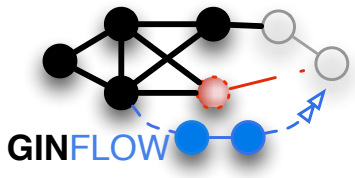
Why another workflow execution manager?



GinFlow does not ...

- intend to compete with existing WMSs
- address a specific scientific area

Why another workflow execution manager?



GinFlow does not ...

- intend to compete with existing WMSs
- address a specific scientific area

GinFlow ...

- specifically targets *decentralisation* and *adaptativeness*
- intends to provide a proof of concept / validation playground

Room for improvement in WMS? (1)

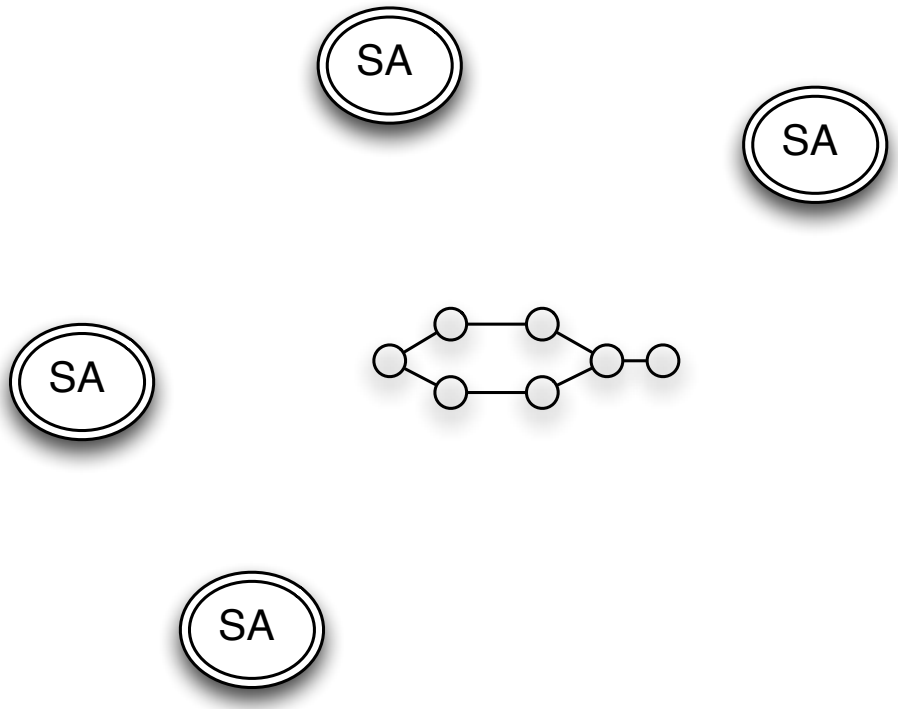
Decentralisation

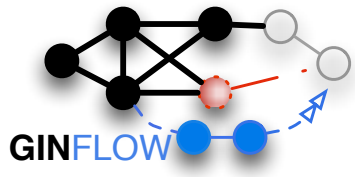
For the sake of:

- removing the need for an orchestrator
- horizontal scaling
- resilience
- **enacting distributed coordination**

GinFlow's decentralised execution

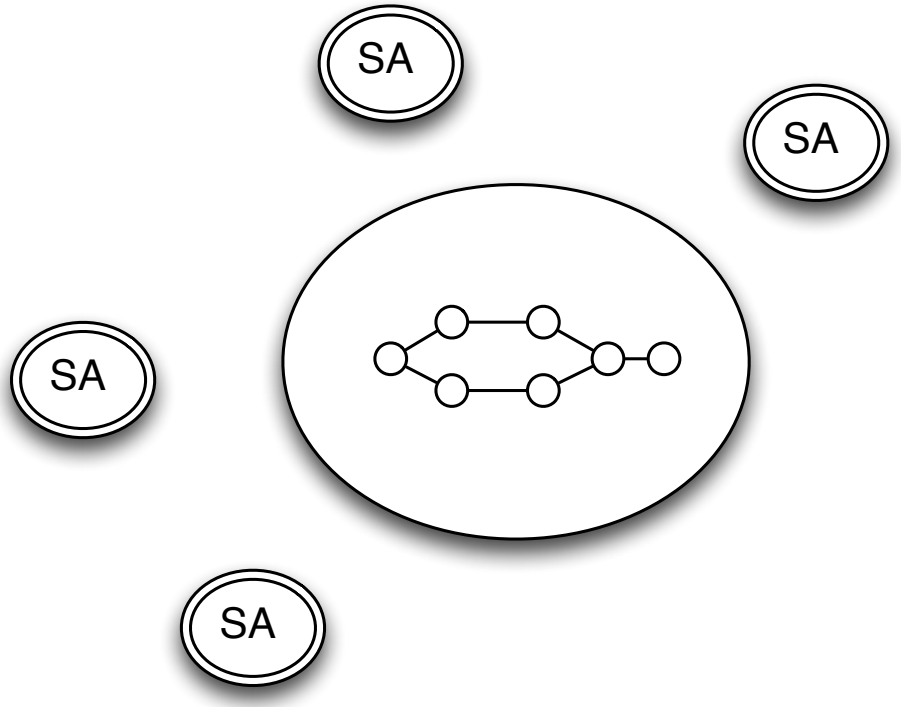
- a set of agents coordinating the execution
- pulling / pushing information into a shared space
- and communicating directly intermediate data

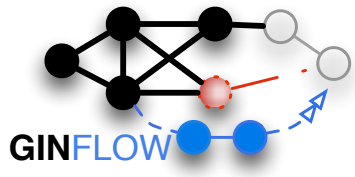




GinFlow's decentralised execution

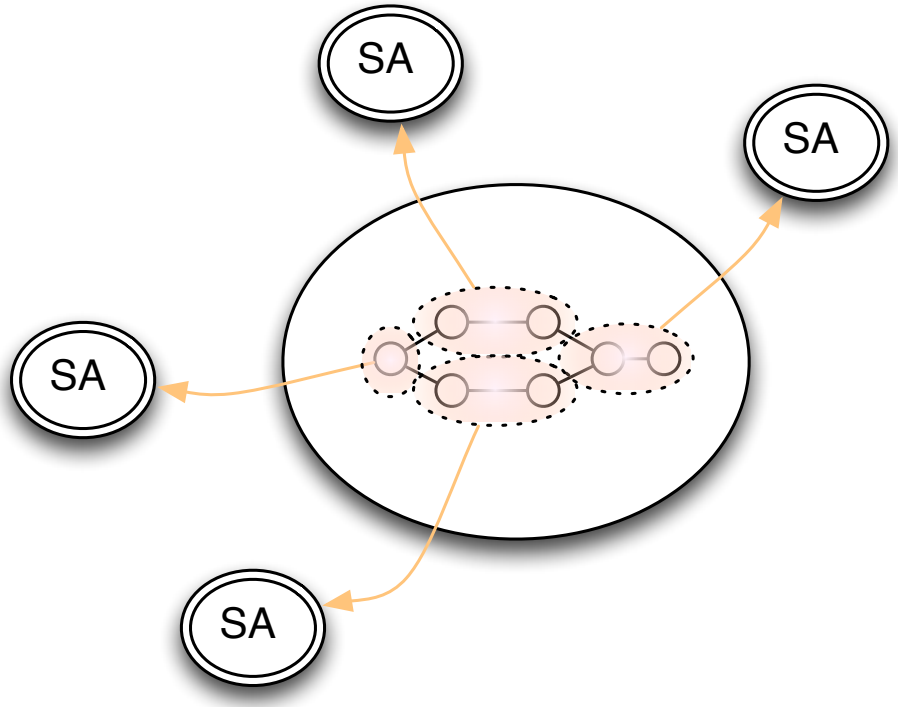
- a set of agents coordinating the execution
- pulling / pushing information into a shared space
- and communicating directly intermediate data

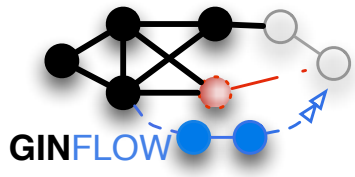




GinFlow's decentralised execution

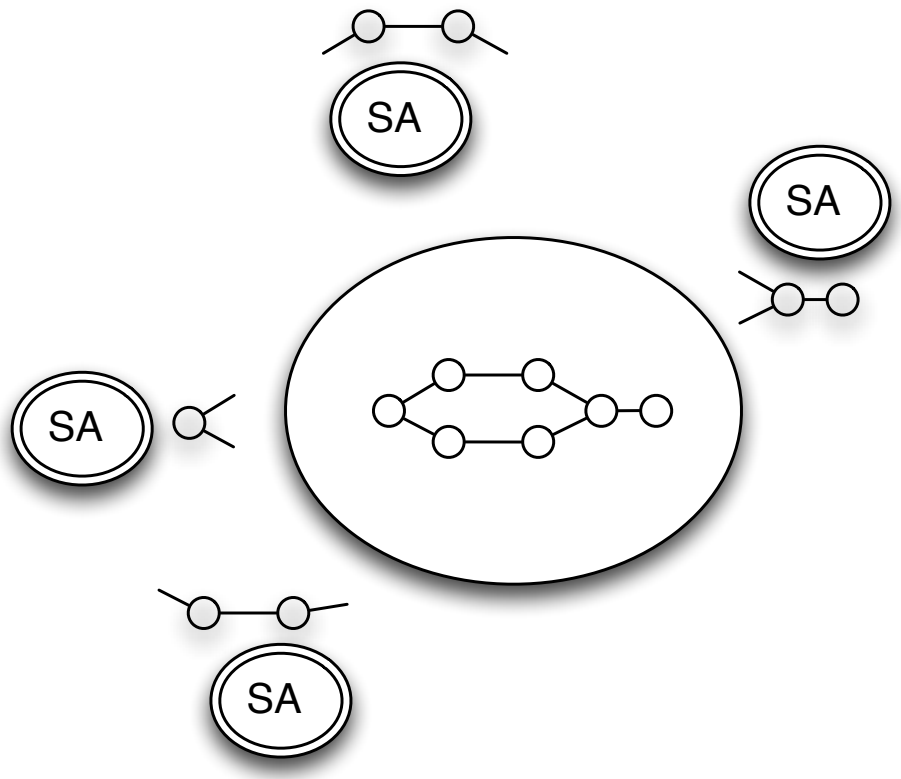
- a set of agents coordinating the execution
- pulling / pushing information into a shared space
- and communicating directly intermediate data

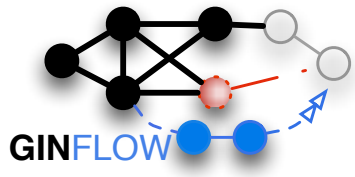




GinFlow's decentralised execution

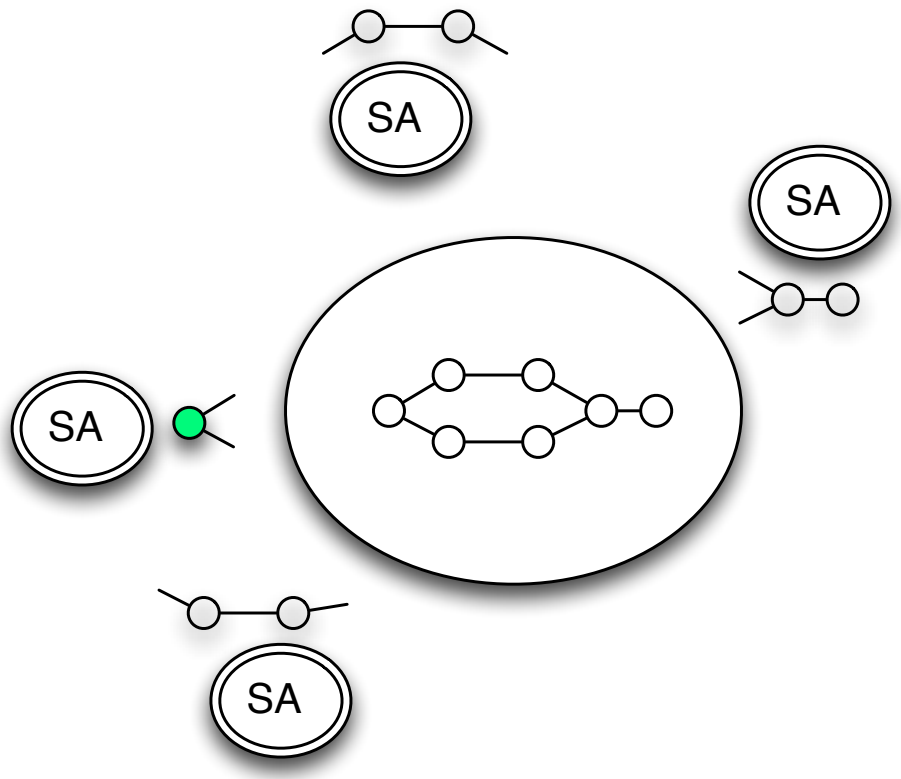
- a set of agents coordinating the execution
- pulling / pushing information into a shared space
- and communicating directly intermediate data

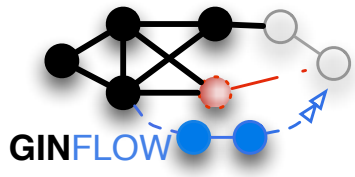




GinFlow's decentralised execution

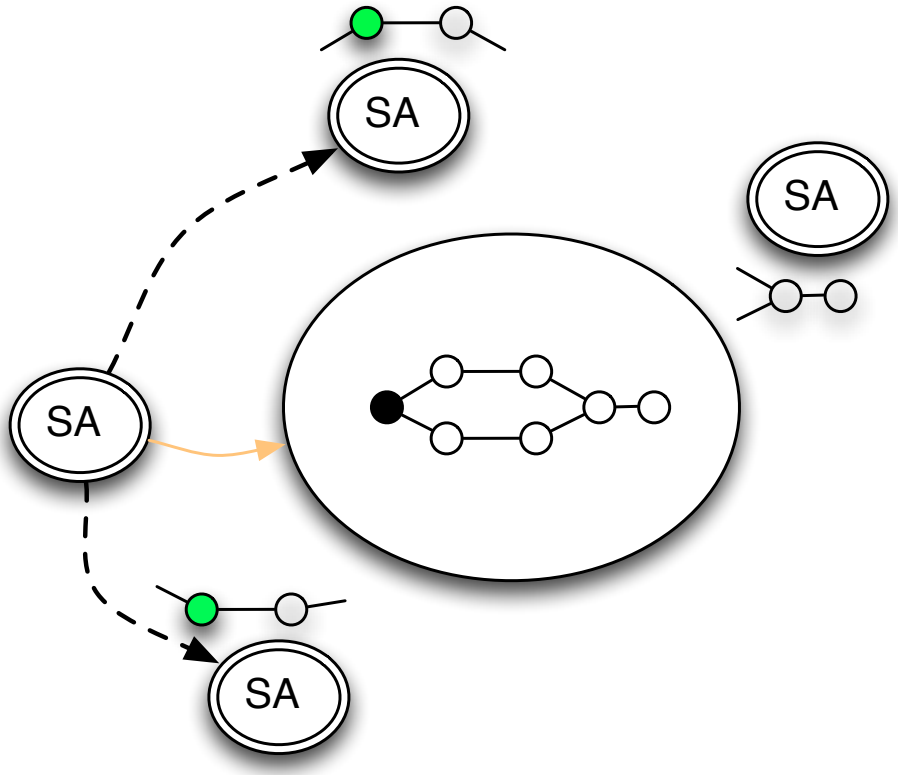
- a set of agents coordinating the execution
- pulling / pushing information into a shared space
- and communicating directly intermediate data

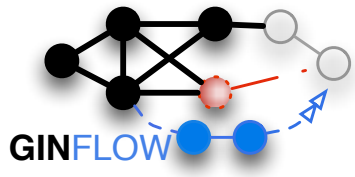




GinFlow's decentralised execution

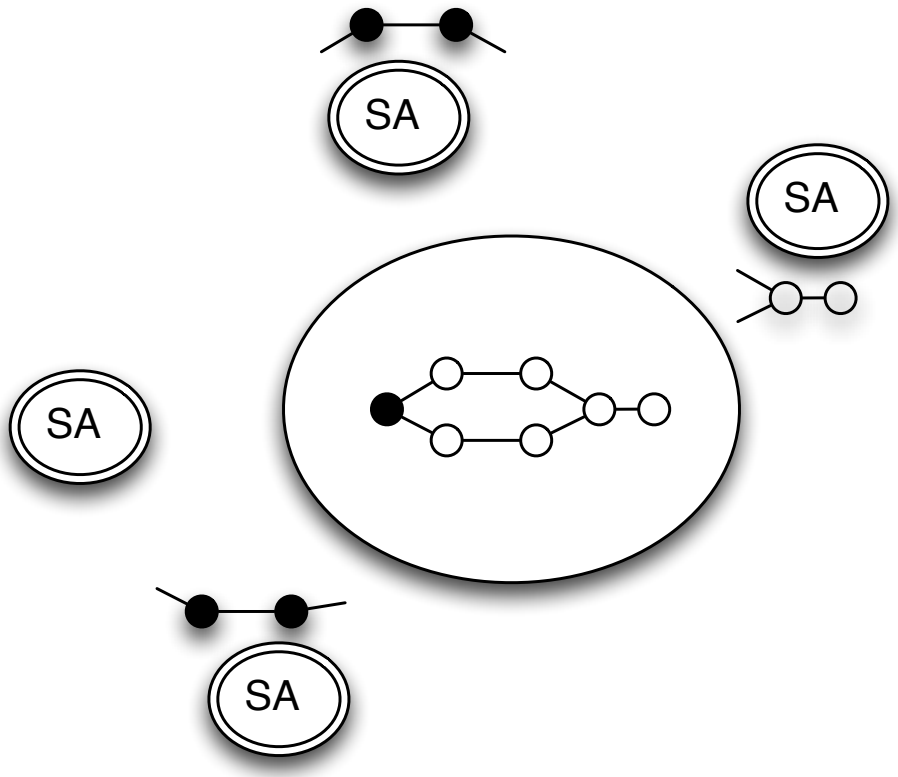
- a set of agents coordinating the execution
- pulling / pushing information into a shared space
- and communicating directly intermediate data

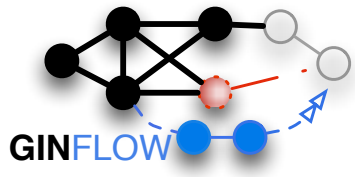




GinFlow's decentralised execution

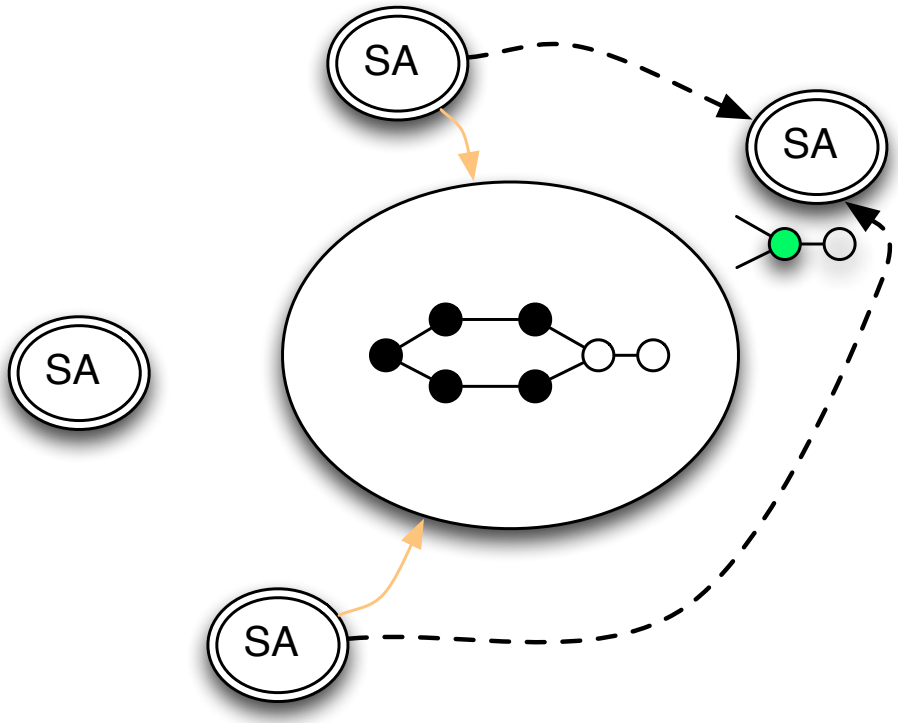
- a set of agents coordinating the execution
- pulling / pushing information into a shared space
- and communicating directly intermediate data

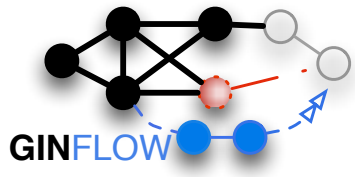




GinFlow's decentralised execution

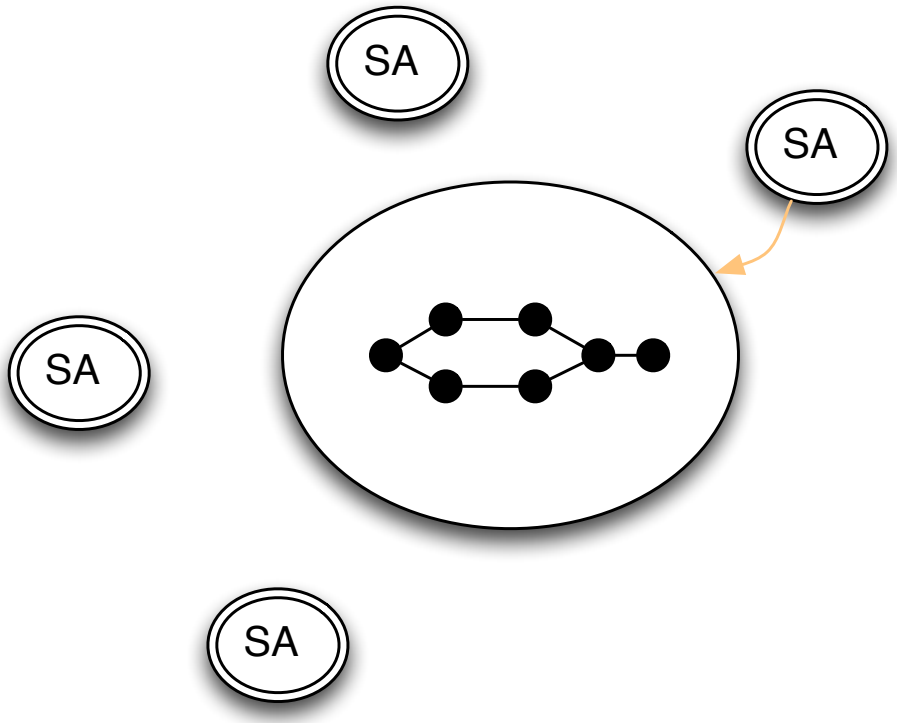
- a set of agents coordinating the execution
- pulling / pushing information into a shared space
- and communicating directly intermediate data

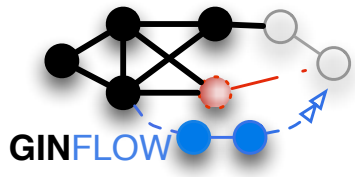




GinFlow's decentralised execution

- a set of agents coordinating the execution
- pulling / pushing information into a shared space
- and communicating directly intermediate data





Room for improvement in WMS? (2)

Adaptiveness

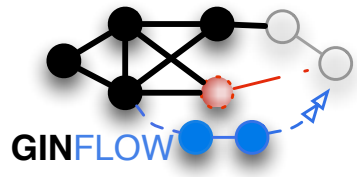
"... most scientific activity consists of exploration of variants and experimentation with alternative settings, which would involve modifying workflows to understand their effects and how to explain those effects. Hence, an important challenge in science is representation of workflow variants, which aims at understanding the impact that a change has on the resulting data products as an aid to scientific discourse."[a]

[a] Gil. et al., Examining the Challenges of Scientific Workflows. IEEE Computer, 2007

Type of adaptiveness addressed by GinFlow

Dynamic adaptation of the workflow shape

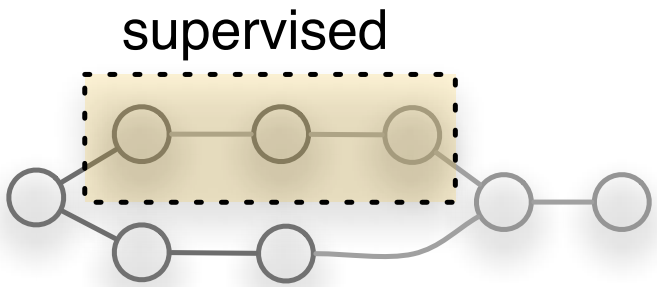
- guided by the user which tags some portion as *supervised*
- and either predefines an alternative
- or (partially) redesigns the workflow after some failure

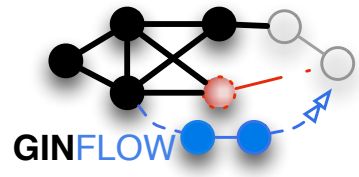


Type of adaptiveness addressed by GinFlow

Dynamic adaptation of the workflow shape

- guided by the user which tags some portion as *supervised*
- and either predefines an alternative
- or (partially) redesigns the workflow after some failure

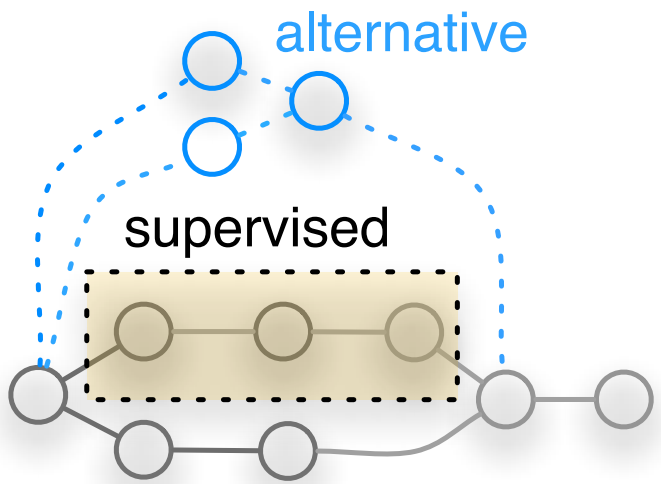


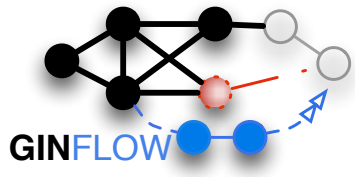


Type of adaptiveness addressed by GinFlow

Dynamic adaptation of the workflow shape

- guided by the user which tags some portion as *supervised*
- and either predefines an alternative
- or (partially) redesigns the workflow after some failure

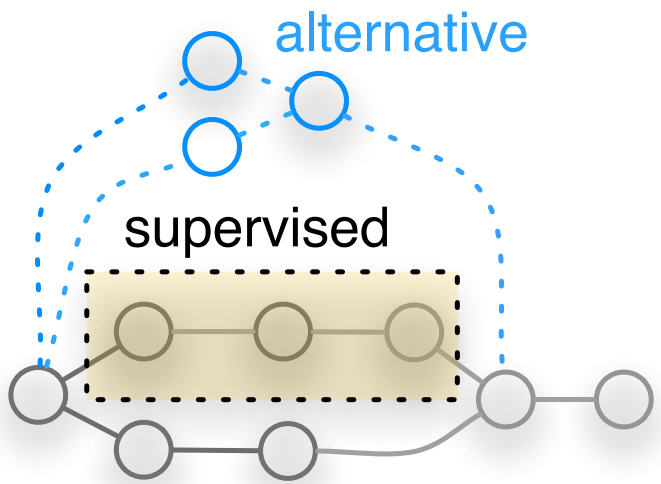


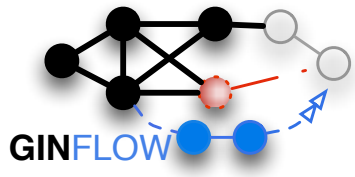


Type of adaptiveness addressed by GinFlow

Dynamic adaptation of the workflow shape

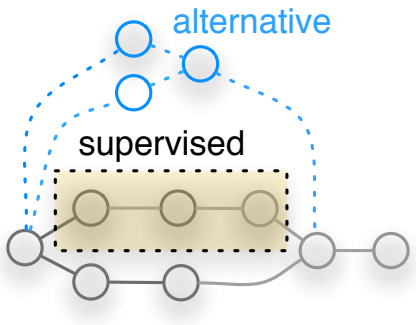
- guided by the user which tags some portion as *supervised*
- and either predefines an alternative
- or (partially) redesigns the workflow after some failure





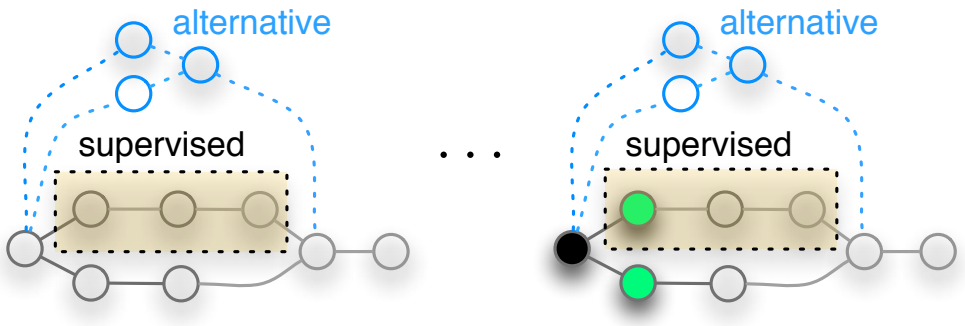
GinFlow's adaptation scenario (1)

Adaptiveness: predefined alternative (*cold adaptation*.)



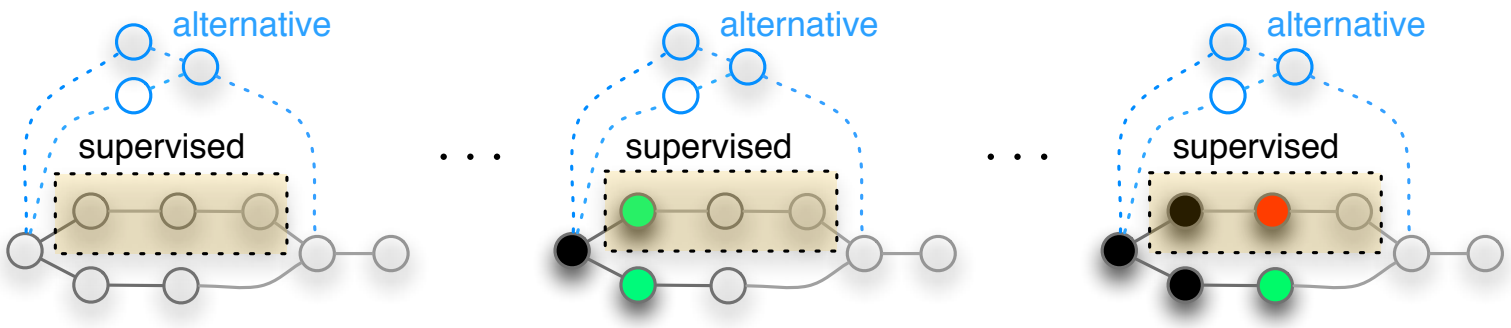
GinFlow's adaptation scenario (1)

Adaptiveness: predefined alternative (*cold adaptation*.)



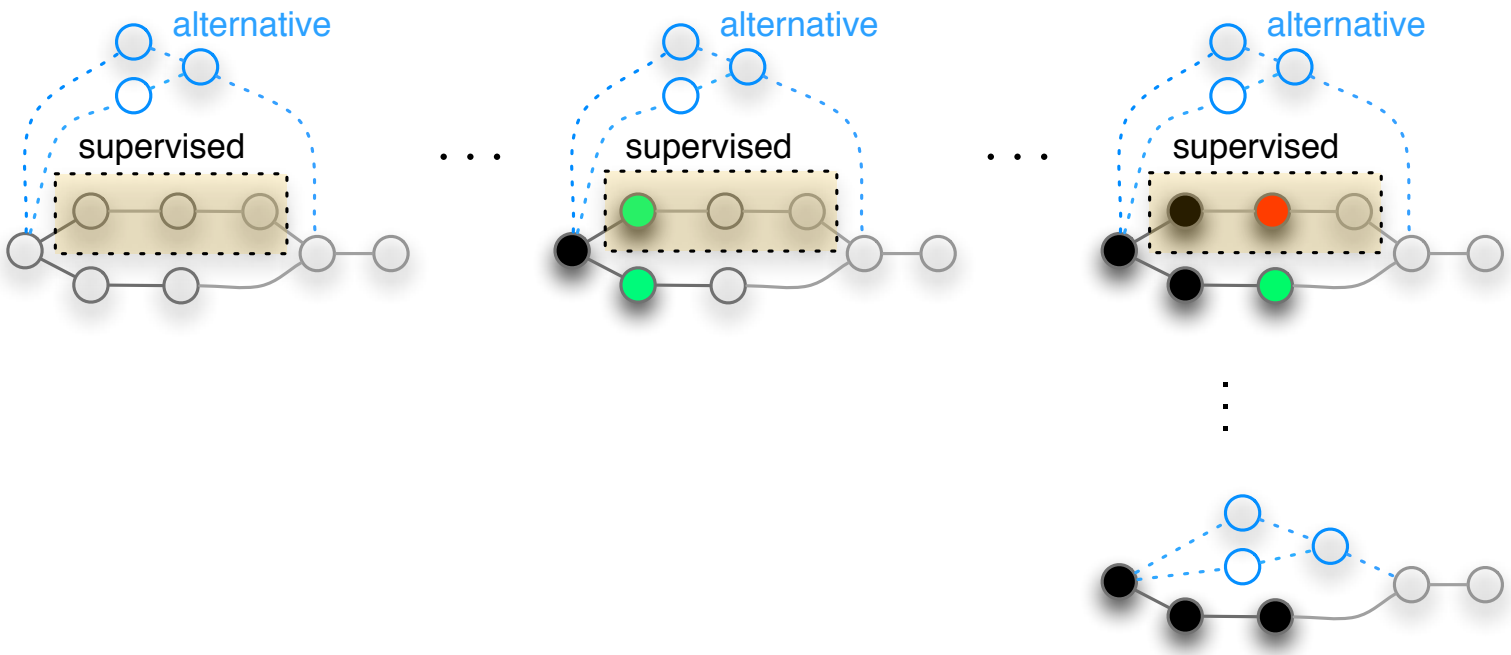
GinFlow's adaptation scenario (1)

Adaptiveness: predefined alternative (*cold adaptation*.)



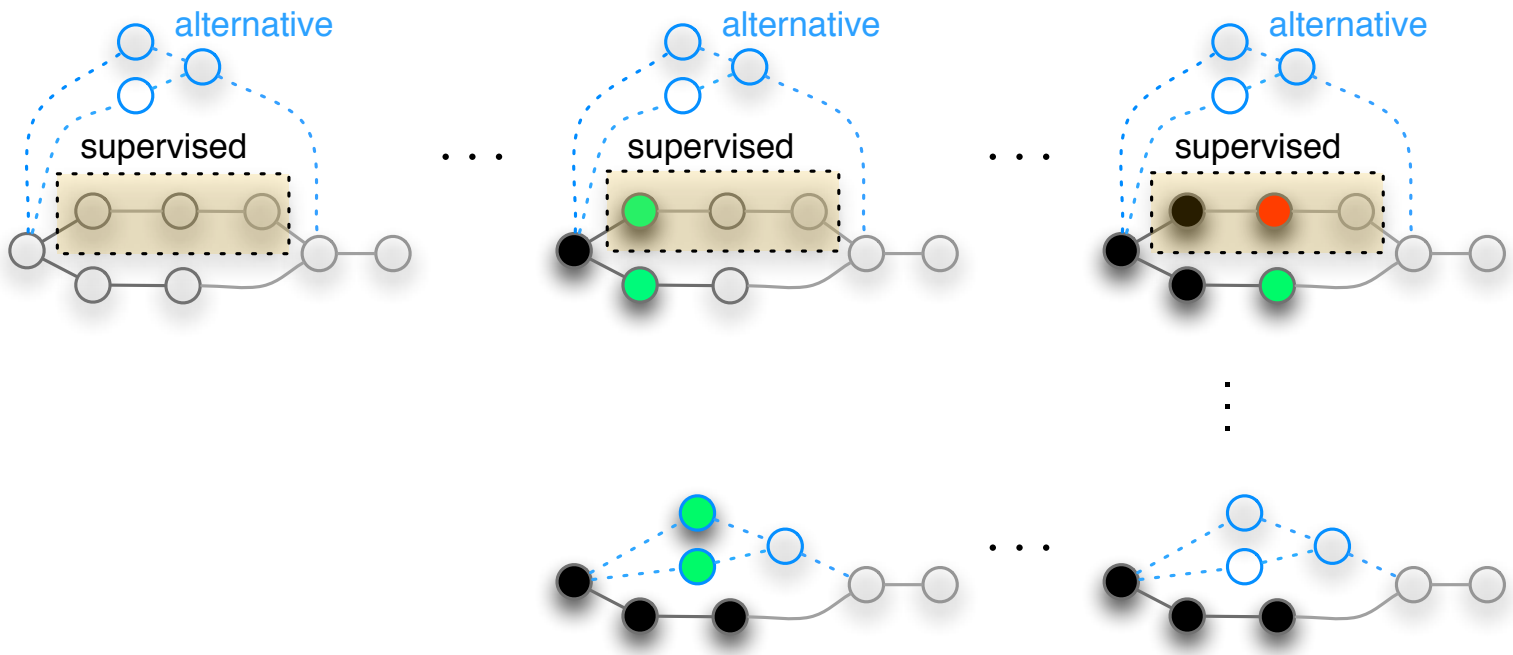
GinFlow's adaptation scenario (1)

Adaptiveness: predefined alternative (*cold adaptation.*)



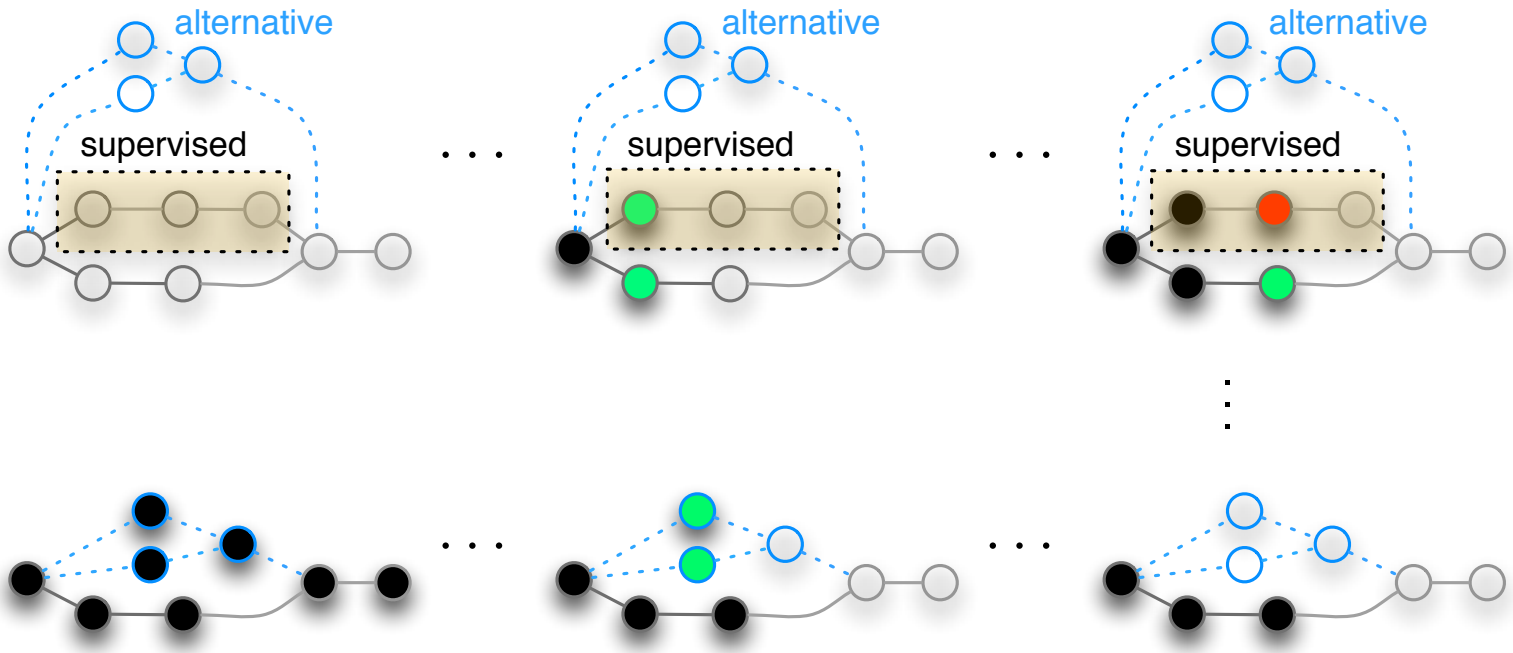
GinFlow's adaptation scenario (1)

Adaptiveness: predefined alternative (*cold adaptation*.)



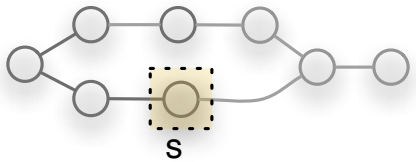
GinFlow's adaptation scenario (1)

Adaptiveness: predefined alternative (*cold adaptation*.)



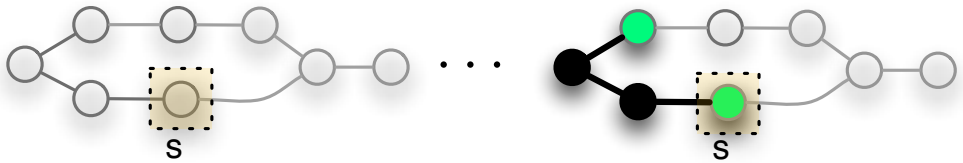
GinFlow's adaptation scenario (2)

Adaptiveness: partial redesign at run time (*hot adaptation*.)



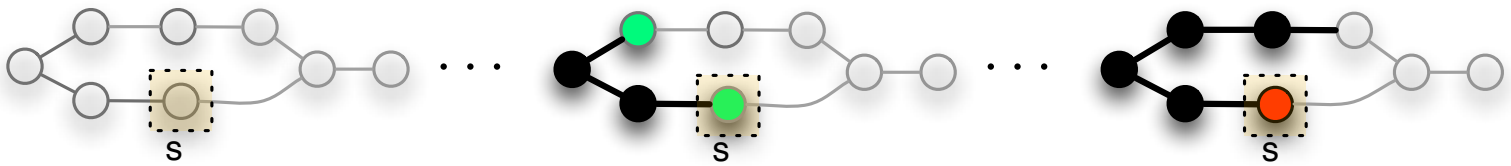
GinFlow's adaptation scenario (2)

Adaptiveness: partial redesign at run time (*hot adaptation*.)



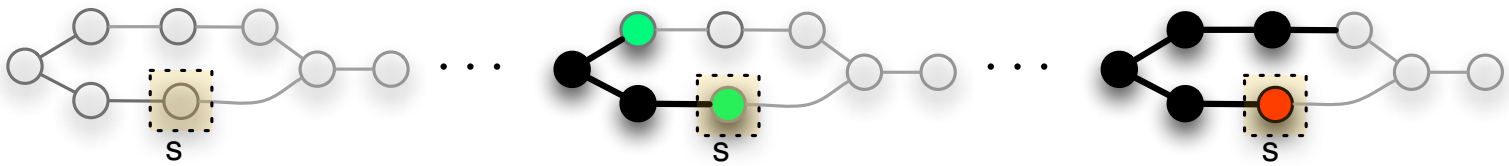
GinFlow's adaptation scenario (2)

Adaptiveness: partial redesign at run time (*hot adaptation*.)



GinFlow's adaptation scenario (2)

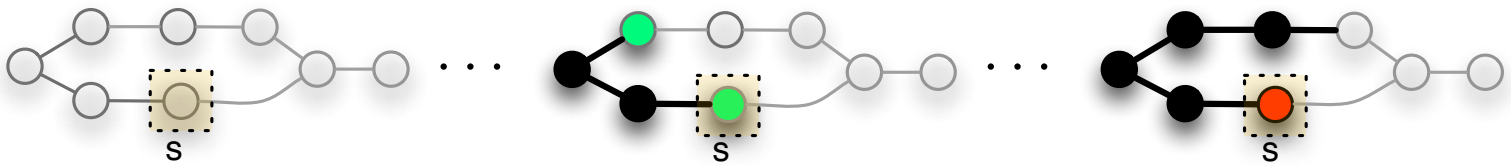
Adaptiveness: partial redesign at run time (*hot adaptation*.)



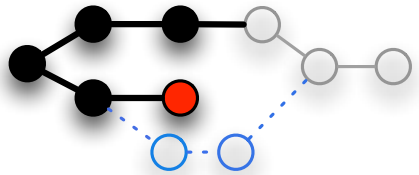
⋮ user intervention

GinFlow's adaptation scenario (2)

Adaptiveness: partial redesign at run time (*hot adaptation*.)

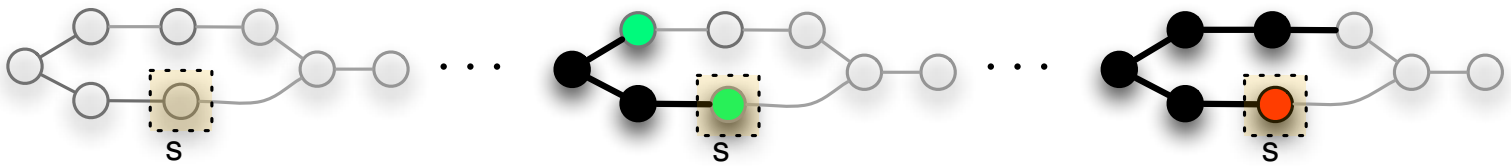


⋮ user intervention

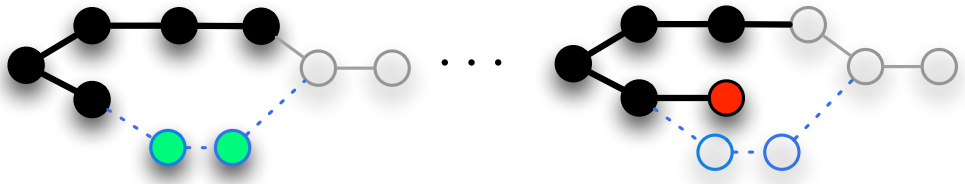


GinFlow's adaptation scenario (2)

Adaptiveness: partial redesign at run time (*hot adaptation*.)

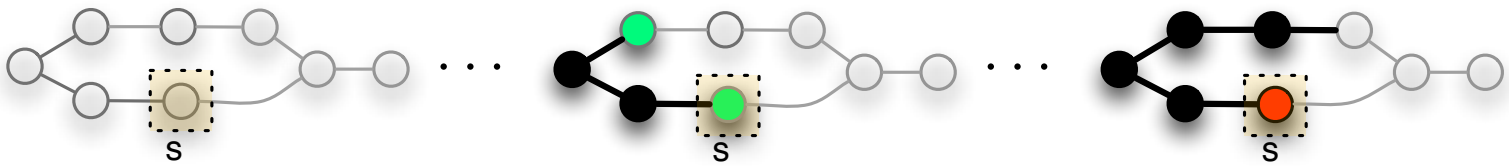


⋮ user intervention

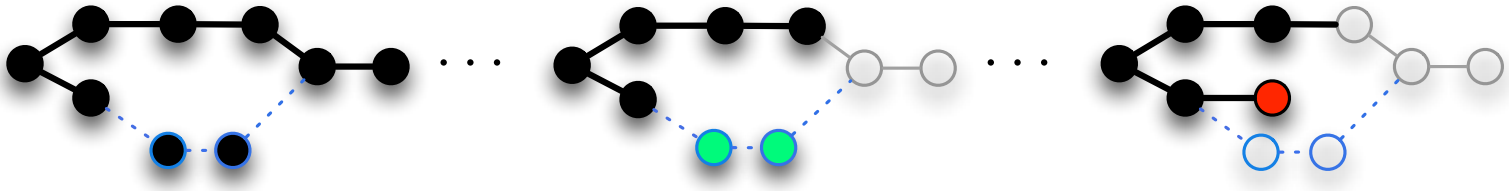


GinFlow's adaptation scenario (2)

Adaptiveness: partial redesign at run time (*hot adaptation*.)



⋮ user intervention



GinFlow's core: a declarative workflow engine

GinFlow's engine was developed using HOCL

- data left unstructured in the multiset
- program as a set of rules
  - identifying patterns in the multiset
  - modifying the multiset
  - concurrently applied by a set of service agents

GinFlow's core: a declarative workflow engine

GinFlow's engine was developed using HOCL

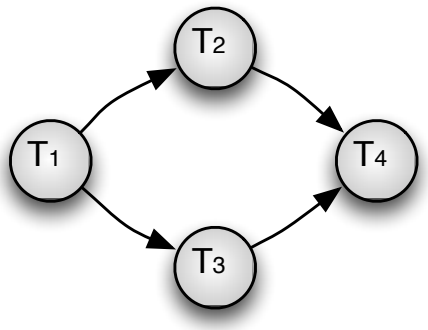
- data left unstructured in the multiset : **the workflow**
- program as a set of rules
  - identifying patterns in the multiset
  - modifying the multiset
  - concurrently applied by a set of service agents

GinFlow's core: a declarative workflow engine

GinFlow's engine was developed using HOCL

- data left unstructured in the multiset : **the workflow**
- program as a set of rules : **the workflow engine**
  - identifying patterns in the multiset
  - modifying the multiset
  - concurrently applied by a set of service agents

Specification of a workflow



$$
\begin{aligned}
&\langle \\
&\quad T_1 : \langle \text{SRC} : \langle\rangle, \text{DST} : \langle T_2, T_3\rangle, \text{SRV} : \texttt{s1}, \text{IN} : \langle \texttt{input}\rangle\rangle, \\
&\quad T_2 : \langle \text{SRC} : \langle T_1\rangle, \text{DST} : \langle T_4\rangle, \text{SRV} : \texttt{s2}, \text{IN} : \langle\rangle\rangle, \\
&\quad T_3 : \langle \text{SRC} : \langle T_1\rangle, \text{DST} : \langle T_4\rangle, \text{SRV} : \texttt{s3}, \text{IN} : \langle\rangle\rangle, \\
&\quad T_4 : \langle \text{SRC} : \langle T_2, T_3\rangle, \text{DST} : \langle\rangle, \text{SRV} : \texttt{s4}, \text{IN} : \langle\rangle\rangle \\
&\rangle
\end{aligned}
$$

Each line is sent and locally stored on the engine responsible for the task.
Still, we need rules to execute the workflow (and modify its state)

Generic rules to execute the workflows

Rules duplicated on each SA to:

- start the task (once all incoming dependencies are satisfied)
- move data between tasks (once a result has been produced)

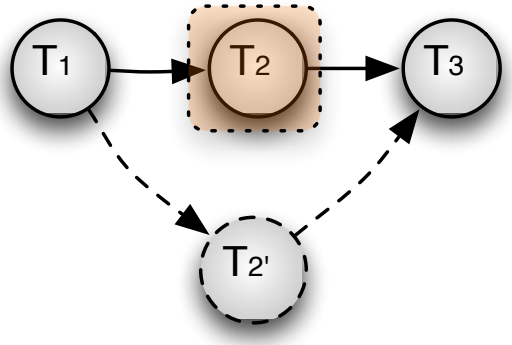
Generic rules to execute the workflows

Rules duplicated on each SA to:

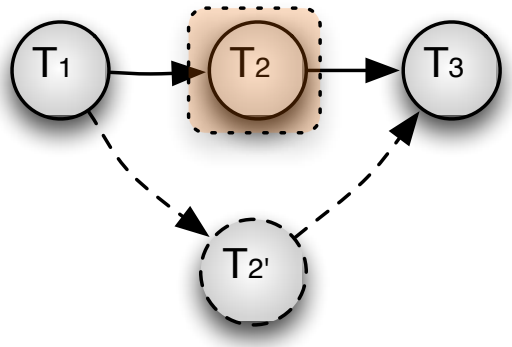
- start the task (once all incoming dependencies are satisfied)
- move data between tasks (once a result has been produced)

$$
\begin{aligned}
\text{gw\_setup} = \quad & \textbf{replace} \quad \text{SRC} : \langle\rangle, \text{IN} : \langle\omega\rangle \\
& \textbf{by} \quad \text{SRC} : \langle\rangle, \text{PAR} : \mathsf{list}(\omega) \\
\text{gw\_call} = \quad & \textbf{replace} \quad \text{SRC} : \langle\rangle, \text{SRV} : \mathtt{s}, \text{PAR} : \ell_{\text{PAR}}, \text{RES} : \langle\omega\rangle \\
& \textbf{by} \quad \text{SRC} : \langle\rangle, \text{SRV} : \mathtt{s}, \text{RES} : \langle \mathtt{invoke}(\mathtt{s}, \ell_{\text{PAR}}), \omega\rangle \\
\text{gw\_pass} = \quad & \textbf{replace} \quad \text{RES} : \langle\omega_{\text{RES}}\rangle, \text{DST} : \langle T_j, \omega_{\text{DST}}\rangle \\
& \textbf{by} \quad \text{RES} : \langle\omega_{\text{RES}}\rangle, \text{DST} : \langle\omega_{\text{DST}}\rangle, \mathtt{send}(\omega_{RES}, T_j)
\end{aligned}
$$

Specification of an (abstract) adaptive workflow



Specification of an (abstract) adaptive workflow



2.01 $\langle$

2.02 $\quad T_1 : \langle \text{SRC} : \langle\rangle, \text{DST} : \langle T_2 \rangle, \text{SRV} : \texttt{s1}, \text{IN} : \langle \texttt{input} \rangle \rangle,$

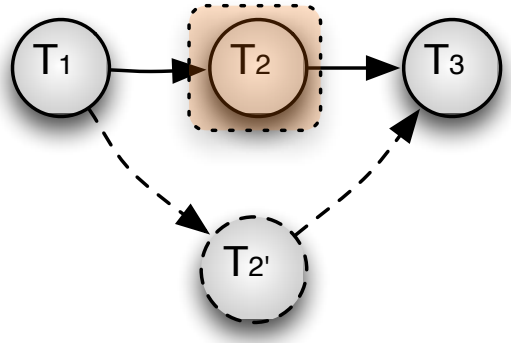
2.03 $\quad T_2 : \langle \text{SRC} : \langle T_1 \rangle, \text{DST} : \langle T_3 \rangle, \text{SRV} : \texttt{s2}, \text{IN} : \langle\rangle \rangle,$

2.04 $\quad T_3 : \langle \text{SRC} : \langle T_2 \rangle, \text{DST} : \langle T_4 \rangle, \text{SRV} : \texttt{s3}, \text{IN} : \langle\rangle \rangle,$

2.05 $\quad T_{2'} : \langle \text{SRC} : \langle T_1 \rangle, \text{DST} : \langle T_3 \rangle, \text{SRV} : \texttt{s2'}, \text{IN} : \langle\rangle \rangle$

2.06 $\rangle$

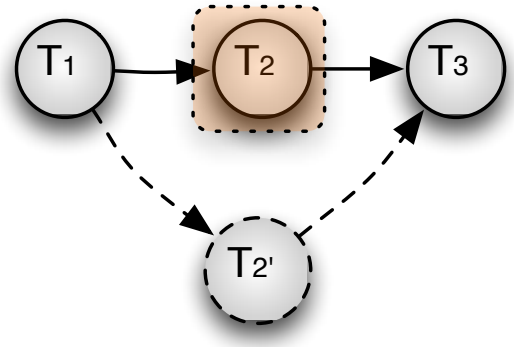
Generation of specific adaptation rules



Specific rules to:

- trigger adaptation
- spread new information where needed
- adapt locally with the new information

Generation of specific adaptation rules

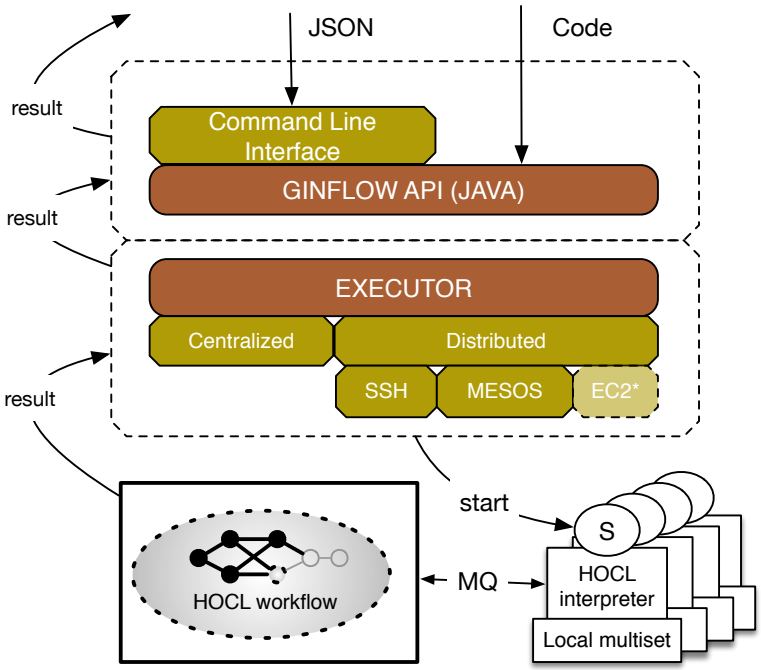


Specific rules to:

- trigger adaptation
- spread new information where needed
- adapt locally with the new information

$$\begin{aligned}
\text{trigger_adapt}_{T2} = \quad & \textbf{replace} \quad \text{RES} : \langle \text{ERROR} \rangle \quad \textbf{by} \quad \texttt{send}(\text{ADAPT}, [T_1, T_3]) \\
\text{update_dst}_{T1} = \quad & \textbf{replace} \quad \text{DST} : \langle \rangle, \text{ADAPT} \quad \textbf{by} \quad \text{DST} : \langle T_{2'} \rangle \\
\text{update_src}_{T3} = \quad & \textbf{replace} \quad \text{SRC} : \langle \omega_{SRC} \rangle, \text{IN} : \langle \omega_{IN} \rangle, \text{ADAPT} \\
& \textbf{by} \quad \text{SRC} : \langle \omega_{SRC}, T_{2'} \rangle, \text{IN} : \langle \rangle
\end{aligned}$$

Towards GinFlow



- Development
  - Refactoring of the initial prototype
  - Extension to different MQs
  - Executors (deployment)
  - APIs (Java, JSON, GUI)
- Validation over Grid'5000
- Open-source release
- Binding with TIGRES (LBL)

L3 Antoine Chatalic

Gossip-based molecule dissemination

The best-effort way

Revisiting the model's complexity

Testing the model's viability at large scale

Going distributed

Collaboration with Marin Bertier

PhD Marko Obrovac

Distributed atomic capture

Higher-order

M2 Matthieu Perrin

The idealist way (perfect but costly)

Distributed discovery and inertia detection

Software protoype

Post-doc Javier Rojas Balderrama

Chemically-inspired workflow reconfiguration

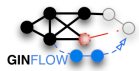
EA DALHIS

Integration into Tigres (LBL)

Exploring chemically-inspired workflow management

Workflow management system

ANR AutoChem



ADT GinFlow Matthieu Simonin

An architecture for decentralized workflow execution

PhD Héctor Fernandez

L3 Antoine Chatalic

Gossip-based molecule dissemination

The best-effort way

Revisiting the model's complexity

Testing the model's viability at large scale

Going distributed

Collaboration with Marin Bertier

PhD Marko Obrovac

Distributed atomic capture

Higher-order

M2 Matthieu Perrin

IJFCS

The idealist way (perfect but costly)

Distributed discovery and inertia detection

Software protoype

ICDCN 2012

ICDCN 2013

JPDC

IJUC

DAIS 2013

CMC 2015

Post-doc Javier Rojas Balderrama

Chemically-inspired workflow reconfiguration

EA DALHIS

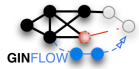
Integration into Tigres (LBL)

WORKS@SC 2014

Exploring chemically-inspired workflow management

Workflow management system

ANR AutoChem



IPDPS 2016

ADT GinFlow Matthieu Simonin

ICWS 2010

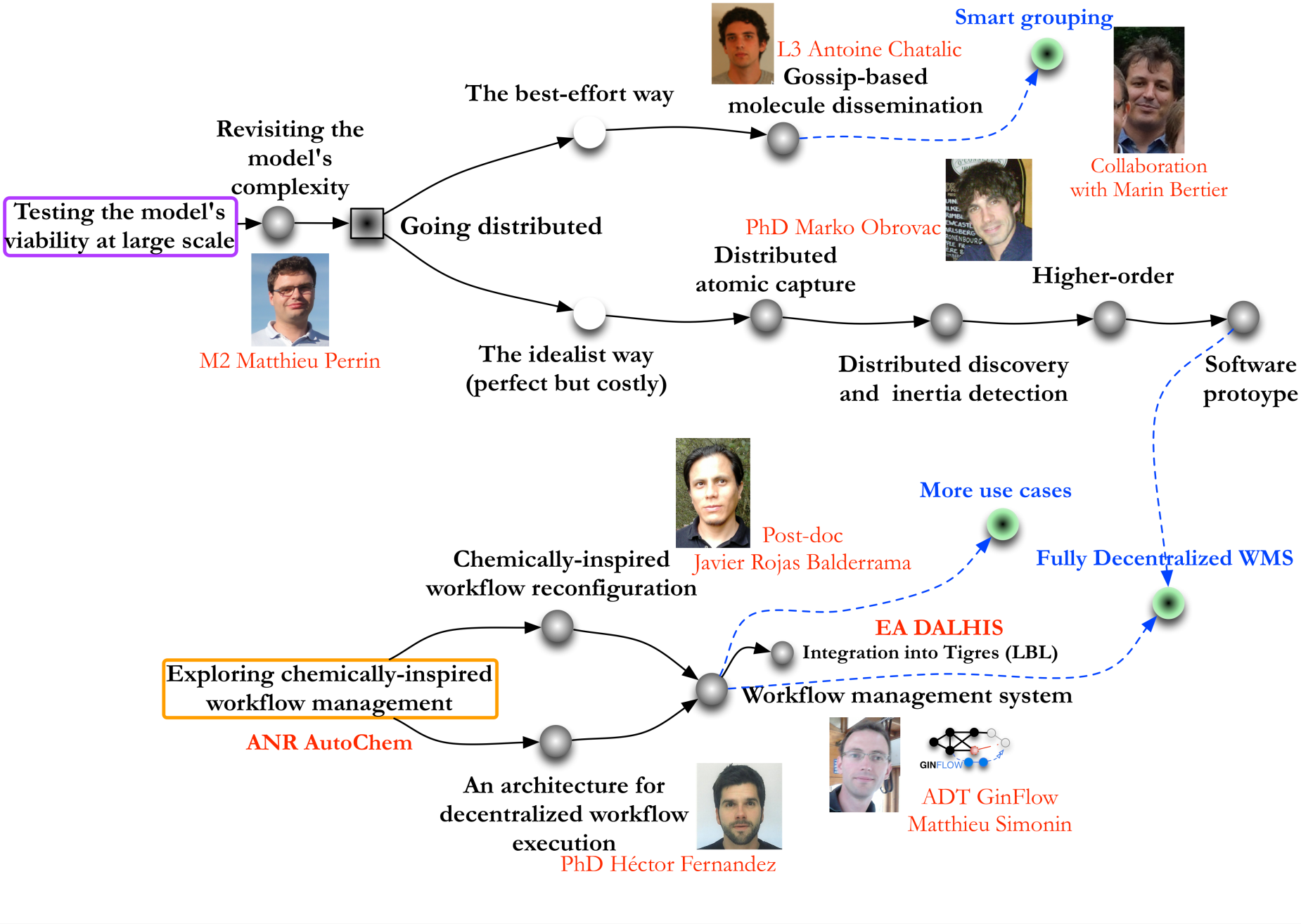
eScience 2011

TSC

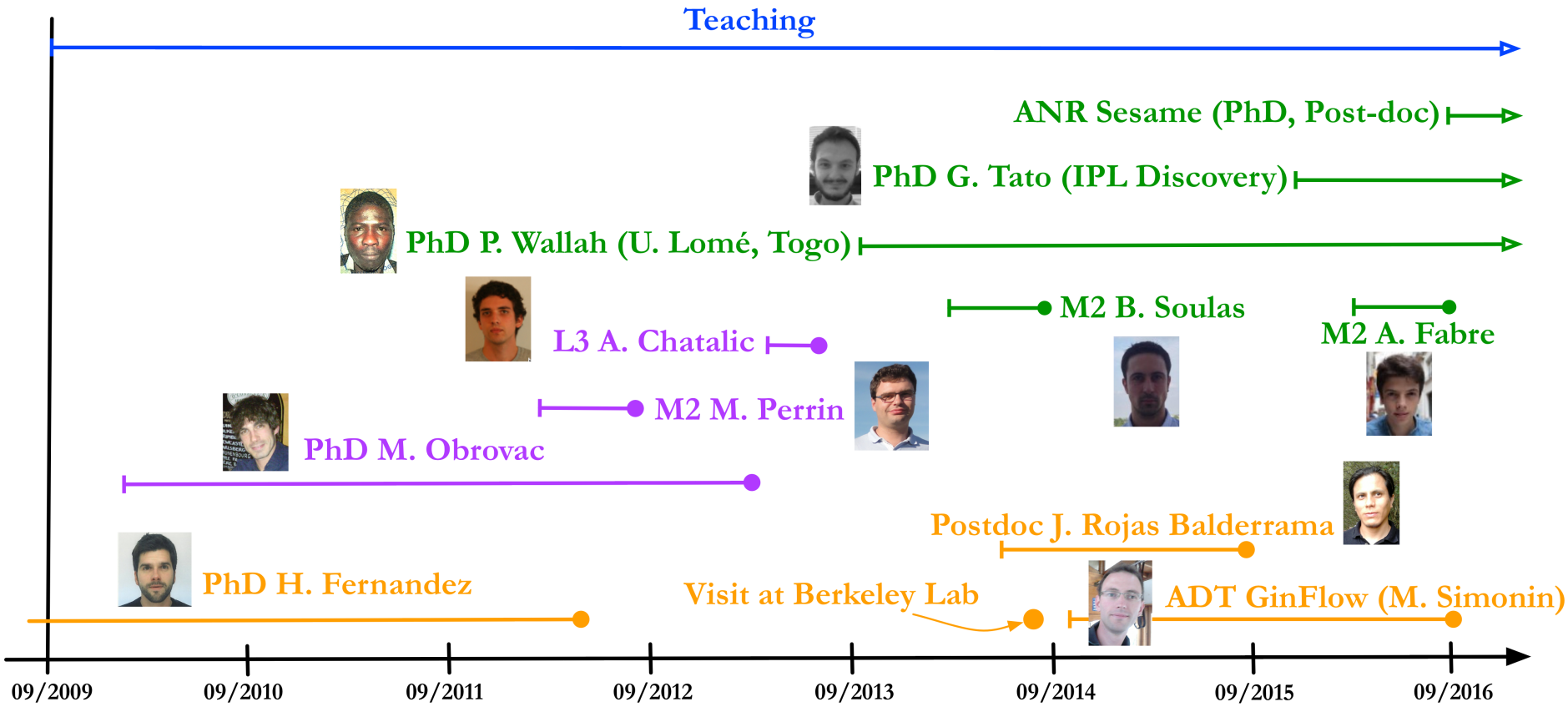
FGCS

An architecture for decentralized workflow execution

PhD Héctor Fernandez


Testing the model's viability at large scale
Revisiting the model's complexity
M2 Matthieu Perrin
Going distributed
The best-effort way
L3 Antoine Chatalic
Gossip-based molecule dissemination
Smart grouping
Collaboration with Marin Bertier
PhD Marko Obrovac
Distributed atomic capture
Higher-order
The idealist way (perfect but costly)
Distributed discovery and inertia detection
Software protoype
Post-doc
Javier Rojas Balderrama
More use cases
Chemically-inspired workflow reconfiguration
Fully Decentralized WMS
EA DALHIS
Integration into Tigres (LBL)
Exploring chemically-inspired workflow management
ANR AutoChem
Workflow management system
An architecture for decentralized workflow execution
PhD Héctor Fernandez
GINFLOW
ADT GinFlow
Matthieu Simonin

Timeline


Teaching
ANR Sesame (PhD, Post-doc)
PhD G. Tato (IPL Discovery)
PhD P. Wallah (U. Lomé, Togo)
M2 B. Soulas
M2 A. Fabre
L3 A. Chatalic
M2 M. Perrin
PhD M. Obrovac
Postdoc J. Rojas Balderrama
PhD H. Fernandez
Visit at Berkeley Lab
ADT GinFlow (M. Simonin)
09/2009
09/2010
09/2011
09/2012
09/2013
09/2014
09/2015
09/2016

Thx.