A Self-Organising Infrastructure for Chemical-Semantic Coordination

Elena Nardini   Mirko Viroli
Matteo Casadei   Andrea Omicini

ALMA MATER STUDIORUM – Università di Bologna
{elena.nardini,mirko.viroli,m.casadei,andrea.omicini}@unibo.it

WOA 2010

Rimini, Italy
September 7, 2010

Outline

Chemical Tuple Spaces

Chemical tuple spaces for modelling interaction rules in ecosystems.

Main idea

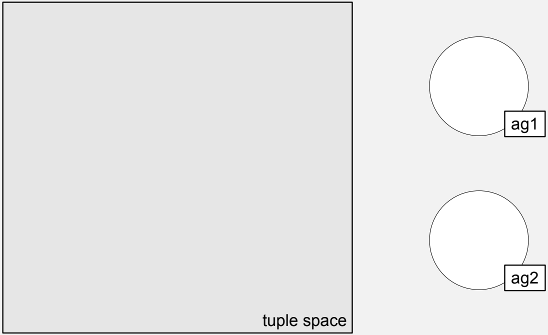
- Tuple spaces + chemical-like reactions as coordination laws
- Tuples have a concentration (a.k.a. weight, or activity value)
- Concentration is evolved "exactly" as in chemistry [4]
- Some reactions can even fire a tuple from one space to another

Why designing coordination with this chemical metaphor?

- Chemistry fits coordination (Gamma)
- Can get inspiration from natural/artificial (bio)chemistry
- Can model population dynamics (prey-predator [2])

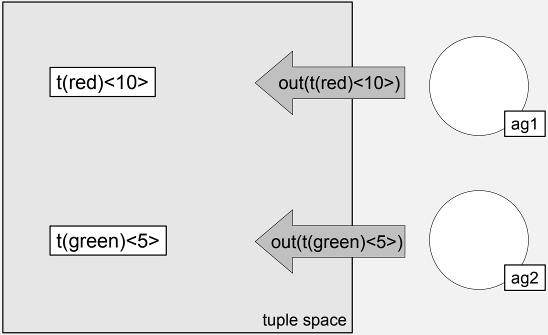
First settings

One tuple space, two agents



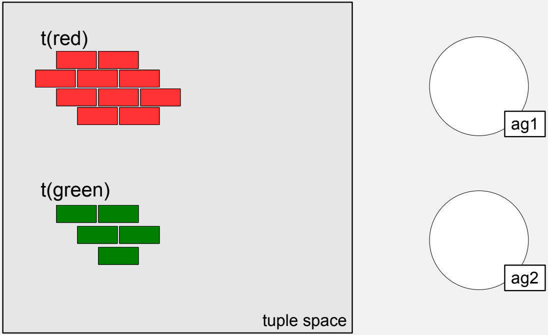
Inserting tuples

Primitive out: default concentration is 1



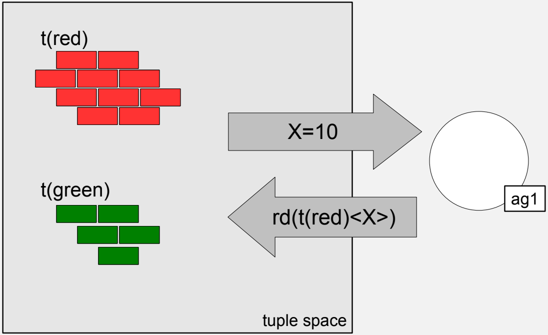
A pictorial representation

A tuple as substance of uniform molecules – still a single tuple



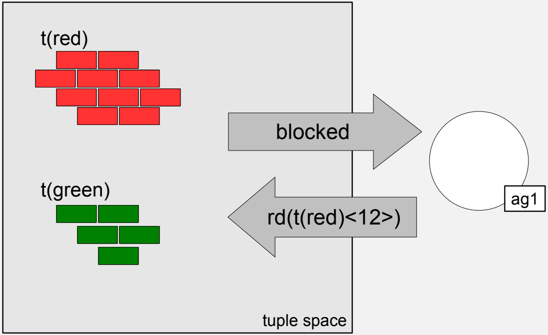
Reading Tuples

Primitive `rd`: reading current concentration



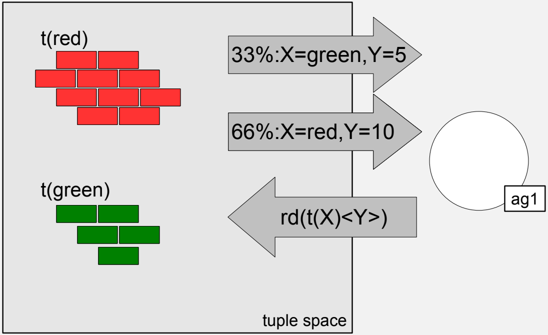
Reading Tuples

Primitive `rd`: reading a given amount – possibly blocking



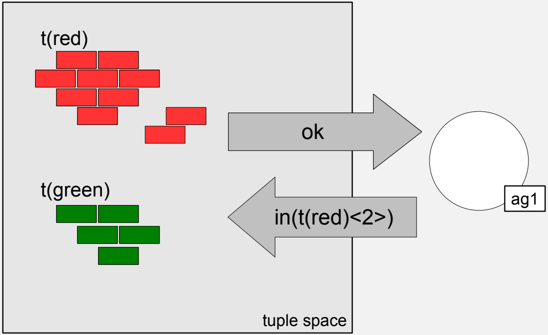
Reading Tuples

Primitive `rd`: concentration as probability, i.e., relevance



Removing Tuples

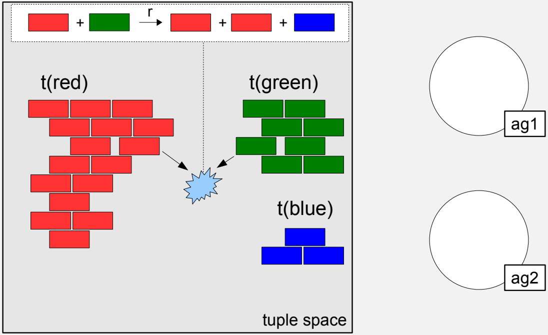
Primitive `in`: removing entirely or partially a tuple



Installing Chemical Reactions

A chemical reaction, with tuples in place of molecules

$$\mathtt{t(red)} + \mathtt{t(green)} \xrightarrow{r} \mathtt{t(red)} + \mathtt{t(red)} + \mathtt{t(blue)}$$

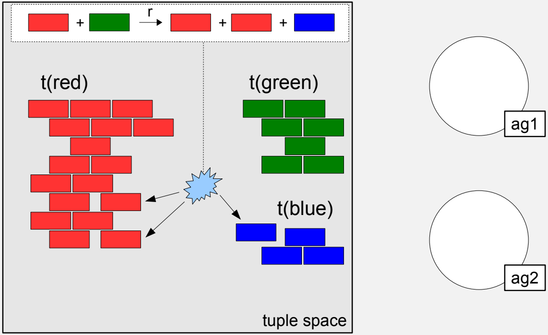


Firing Chemical Reactions

Reactions are executed over time according to [4]

$$\mathtt{t(red)} + \mathtt{t(green)} \xrightarrow{r} \mathtt{t(red)} + \mathtt{t(red)} + \mathtt{t(blue)}$$

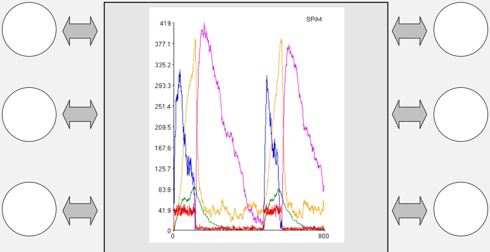
Transition (Markovian) rate: $r * \#\mathtt{t(red)} * \#\mathtt{t(green)}$



A tuple space as a chemical solution

Coordination through an exact chemical solution of tuples

- The tuple space resembles a chemical solution in a glass
- Each tuple resembles a chemical substance
- Agents observe, insert and remove substances
- Tuple concentration drives the selection of chemical reactions



biochemical tuple space (circadian clock)

On matching and rates

Chemical reactions

- are general and not specific for particular chemical substances
  - $\rightarrow$ as tuples, they are expressed in terms of templates
    Example: $\mathtt{s(X)} \xrightarrow{\mathtt{r}} 0$ and $\mathtt{s(X)\ +\ r(Y)} \xrightarrow{\mathtt{r}} \mathtt{s(X)\ +\ s(X)}$
  - $\rightarrow$ the global rate depends from both `r` and the rates of matching substances becomes
    Example: $\mathtt{s(sa)} \xrightarrow{\mathtt{r}} 0$ and $\mathtt{s(sa)\ +\ r(ra)} \xrightarrow{\mathtt{r}} \mathtt{s(sa)\ +\ s(sa)}$
    the rates are respectively $r \times \#sa$ and $r \times \#sa \times \#ra$

On matching and rates

Overcoming syntactic and discrete matching

- We use semantic matching in order to deal with openness requirements [8, 7]
- We use an application-dependent fuzzy match function $\mu(t, t')$ in order to deal with vagueness of information [6]
  - $\mu = 0$ no match, $\mu = 1$ perfect match, $0 < \mu < 1$ partial

$\longrightarrow$ the global rate has to be multiplied with μ

Some implementation facts

How and when selecting a chemical reaction?

Gillespie “direct” simulation algorithm for chemistry [4]

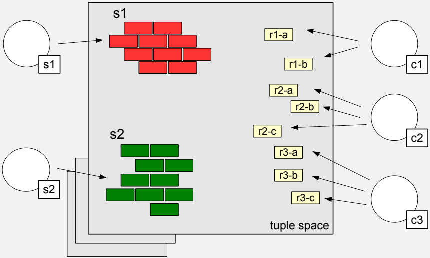
1. Compute the markovian rate $r_1, \ldots, r_n$ of reactions, let R be the sum
2. Choose one of them probabilistically, and execute its transition
3. Proceed again with (1) after $\frac{1}{R} * ln\frac{1}{\tau}$ seconds, with $\tau = random(0, 1)$

Tuple Space implementation

- Can be prototyped on top of TuCSoN
- Tuple centres programmed with the above algorithm
- Also need to implement proper matching

The scenario of service ecosystems

Services and requests as tuples



Clients and services as "individuals of an ecology"

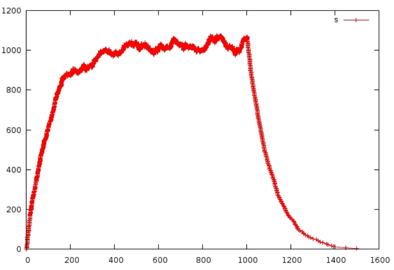
- Unused services fade until completely disappearing
- Concentration of a service increases upon usage
- Similar services compete for survival

Positive-Negative feedback

Service tuples decay, but can be sustained by a feedback token

- Decay rule: $\mathtt{SER} \xrightarrow{r\_dec} 0$
- Feed rule: $\mathtt{publish(SER)} \xrightarrow{r\_feed} \mathtt{publish(SER)} + \mathtt{SER}$

Example simulation: $r\_dec = 0.01, r\_feed = 10$



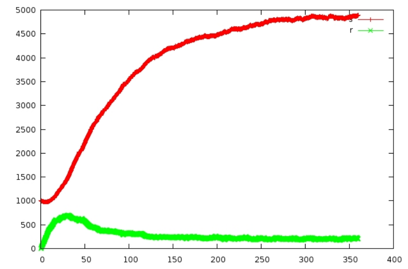
- time 0: Catalyst Token `publish(S)` is inserted
- time 400: Service `S` reaches an equilibrium
- time 1000: The token is removed (or decays)
- time 1600: Service `S` vanishes

Feedback by using (a.k.a. prey-predator)

Idea: Matching Service-Request sustains the service

- Use rule: $\mathtt{SER} + \mathtt{REQ} \xrightarrow{r\_use} \mathtt{SER} + \mathtt{SER} + \mathtt{toserve}(\mathtt{SER}, \mathtt{REQ})$

Sim: $r\_dec = 0.01$, $r\_use = 0.00005$, $request\_arrival\_rate = 50$



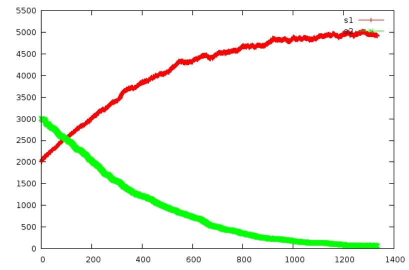
- time 0: Injection of requests raises service level
- time 30: Requests are tamed
- time 350: Unserved requests and service stabilise

Competition

What if more services can handle the same requests?

- higher concentration means higher match frequency
- some service may match better the request, being more proper

Sim: $r\_use_1 = 0.06$, $r\_use_2 = 0.04$



- time 0: The two services are in competition for the same requests
- time 100: The one with better use rate (better match) is prevailing
- time 1300: Service $s2$ lost competition and fades

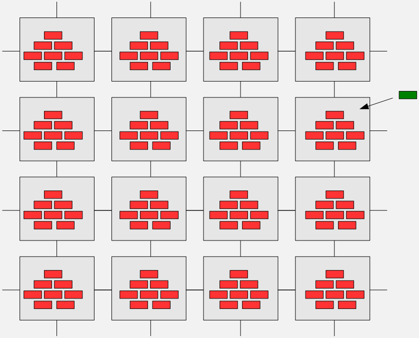
Spatial Diffusion and Competition

One service monopolises a network and its requests

Services continuously diffuse around, by rule:

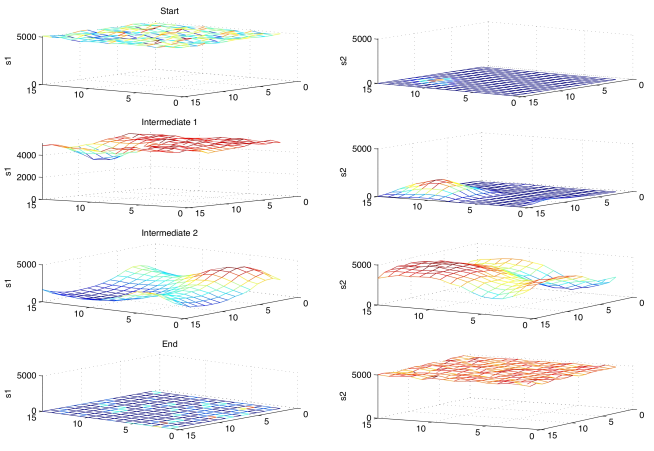
- Diffuse rule: $\mathtt{SER} \xrightarrow{r\_diff} \mathtt{SER}^{\rightsquigarrow}$

Scenario: a better service is injected in a node



Resembling a biological tissue scenario

Example Simulation: $r\_use_1 = 0.05, r\_use_2 = 0.1$



Outline

Chemical tuple spaces in TuCSoN

- Modelling of reactants and laws through tuples
- Realisation of a chemical engine (Gillespie) through ReSpecT reactions
- Realisation of a no syntactic and a no discrete matching through semantic and fuzziness

Chemical reactants and laws

Reactants and laws

- Tuples as reactant individuals in the form `reactant(X,N)`
  - `reactant(sa,1000)` and `reactant(ra,1000)`
- Laws modelling chemical reactions as tuples in the form `law(InputList,Rate,OutputList)`
  - $\mathtt{S} + \mathtt{R} \xrightarrow{10.0} \mathtt{S} + \mathtt{S}$ becomes `law([S,R],10,[S,S])`

Matching and rates

- The law `law([S,R],10,[S,S])` is instantiated with reactants `ra` and `sa` obtaining `law([sa,ra],r1a,[sa,sa])`
- The rate `r1a` is caluclated as $10.0 \times \#as \times \#ar \times \mu(S + R, sa + ra)$

Chemical reactions

Chemical engine as ReSpecT reactions

- Managing chemical laws and reactants
- Controlling engine start and stop
- Choosing the next chemical law to be executed (Gillespie's algorithm)
- Executing chemical laws

Semantic and fuzziness in TuCSoN

A new tuple space model [5]

- *Fuzzy domain ontology* allowing to interpret the semantics associated to the fuzzy knowledge (set of tuples) stored into a tuple space.
- *Fuzzy semantic tuples* fuzzy domain objects – represented by tuples – described so that they can be interpreted in a semantic way, by means of the domain ontology.
- *Fuzzy semantic templates* as fuzzy descriptions of a domain object set.
- *Fuzzy semantic matching* providing the fuzzy domain objects – tuples – described through fuzzy templates.

Semantic and fuzziness in TuCSoN

Fuzzy Description Logic as a formalism

- Fuzzy **SHOIN(D)** Description Logic formalism [1, 5, 6] to describe domain ontologies and objects.
    - Good compromise between expressiveness and complexity.
    - Theoretical counterpart of fuzzy **OWL DL**, that is one of the three species of W3C OWL. OWL being a standard, it well fits the openness requirement.

Semantic and fuzziness in TuCSoN

Tuple space ontology

- Fuzzy OWL DL [6]

Fuzzy semantic tuples

- A fuzzy SHOIN(D)-like language for tuples:

```
Individual ::= iname ':' C
C ::= cname [→ α] | cname '(' R [→ α] ')'
R ::= rname ':' V | rname 'in' '(' Vset ')' | R ',' R
Vset ::= V [→ α] | V [→ α] ',' Vset
V ::= iname | number | string
```

Example:

```
ca : 'CityCar' (hasMaker:ford, hasMaxSpeed:130,
hasColour in (red→ 0.7, black→ 0.3))
```

Semantic and fuzziness in TuCSoN

Fuzzy semantic templates

- A fuzzy SHOIN(D)-like description language for templates:

```
C ::= '$ALL' | '$NONE' | cname | C ',' C |
      C ';' C | '$not' C | D |
      '{' [ iname [→ α] { ',' , iname [→ α] } ] '}' |
      C '(' D ')' | '(' C ')' |
      CW | C'['≥ α']' | C'['≤ α']' |
      DT | M '(' C ')'

CW ::= C→ α | CW + CW

DT ::= crisp'('N','N')' | l'('N','N')' |
      r'('N','N')' | M
```

Semantic and fuzziness in TuCSoN

Fuzzy semantic templates

- A fuzzy SHOIN(D)-like description language for templates:

```
M ::= triangular‘(’N‘,’N‘,’N‘)’ |
      trapezoidal‘(’N‘,’N‘,’N‘,’N‘)’

D ::= F | ‘$exists’ F |‘$all’ F | M

F ::= R ‘in’ C | R ‘:’ I | R ‘:’ I | R ‘:’ Msymb N |
      R ‘:’ ‘=’ string | R

M ::= ‘#’ R Msymb N

R ::= rname | rname ‘/’ vname

Msymb ::= ‘>’ | ‘<’ | ‘≥’ | ‘≤’ | ‘=’
```

Example:

```
‘Car’ ((hasMaxSpeed:280→ 0.6) + (hasMaker:ferrari→ 0.4))
```

Semantic and fuzziness in TuCSoN

Tuple matching

- Fuzzy semantic matching mechanism amounts to look for the fuzzy individuals which are instances of the given fuzzy concept, namely, which tuples match the template.
- Fuzzy Description Logic reasoners like DeLorean [3] can be used to perform this kind of reasoning.

Semantic and fuzziness in TuCSoN

Tuple space primitives

- In a semantic view, tuple space primitives (in, rd, and out) represent the language whereby system components can read, consume, and write knowledge described by means of a domain ontology.
- Each primitive can fail in case of non-consistency with the ontology.

$\Rightarrow$ Differently from the original tuple space semantic, the **out** can fail in case the related tuple is not consistent with the domain ontology.

- In face of a fuzzy template, it is needed too return a fuzzy tuple along with the degree by which it satisfies the template, since this could be different from 1.

Semantic and fuzziness in chemical tuple spaces

Example

We consider a scenario of advertisements for cars:

- Example law: `(SportCar`$\rightarrow$`0.8;(hasMaker:audi`$\rightarrow$`0.2))` $\xrightarrow{\mathtt{r}}$ `0`
- Candidate tuple gives `0.72`:

  `f380:‘SportCar’`$\rightarrow$`0.9(hasMaker : ferrari,`
  `hasMaxSpeed : ’320km/h’, hasColour : red)`
- The global rate of the reaction instance is $r \times \#f380 \times 0.72$

Future work

Further research and development about

- Match factor
- Performance
- Chemical language
- Application cases

Outline

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors.
The Description Logic Handbook: Theory, Implementation, and Applications.
Cambridge University Press, 2003.

Alan A. Berryman.
The origins and evolution of predator-prey theory.
Ecology, 73(5):1530–1535, October 1992.

Fernando Bobillo, Miguel Delgado, and Juan Gomez-Romero.
Delorean: A reasoner for fuzzy OWL 1.1.
In *4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2008)*, volume 423, page 10. CEUR Workshop Proceedings, 2008.

Daniel T. Gillespie.
Exact stochastic simulation of coupled chemical reactions.
The Journal of Physical Chemistry, 81(25):2340–2361, 1977.

Ian Horrocks, Peter F. Patel-Schneider, and Frank Van Harmelen.
From shiq and rdf to owl: The making of a web ontology language.
Journal of Web Semantics, 1:2003, 2003.

Thomas Lukasiewicz and Umberto Straccia.
Managing uncertainty and vagueness in description logics for the semantic web, 2007.

Elena Nardini, Mirko Viroli, and Emanuele Panzavolta.
Coordination in open and dynamic environments with tucson semantic tuple centres.
In *25th Annual ACM Symposium on Applied Computing (SAC 2010)*, volume III, pages 2037–2044, Sierre, Switzerland, 22–26 March 2010. ACM.

Lyndon j. b. Nixon, Elena Simperl, Reto Krummenacher, and Francisco Martin-recuerda.
Tuplespace-based computing for the semantic web: A survey of the state-of-the-art.
Knowl. Eng. Rev., 23(2):181–212, 2008.