Self assembling graphs

Vincent Danos   Fabien Tarissan

PPS, CNRS & Université Paris 7

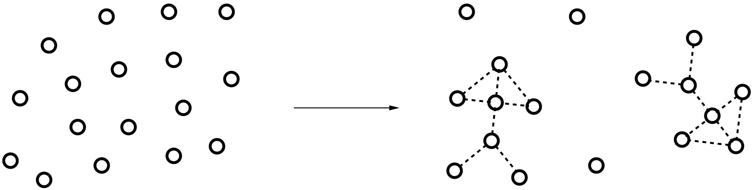
June 15, 2005

The problem

- Question : How a collective behaviour may emerge from elementary interactions (forward and backward)
- Applications
  - Molecular biology (backward)
  - Genetic engineering (forward)
  - Distributed robotics (forward)

The problem

- Question : How a collective behaviour may emerge from elementary interactions (forward and backward)
- Applications
  - Molecular biology (backward)
  - Genetic engineering (forward)
  - Distributed robotics (forward)



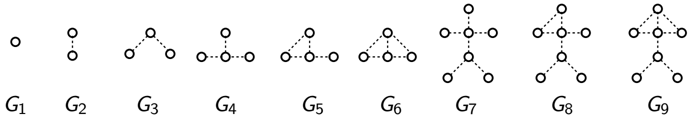
Formal approach

- Model of connections, space, . . .
  What objects are built ?
- Combinatorics of the components:
  What can compute a basic element ?
  Reasonable assumption depends on the context.
- Model of communication:
  How interact an agent ?
  How information is transmitted ?

We look for a syntaxe able to describe the elements and the assembling.

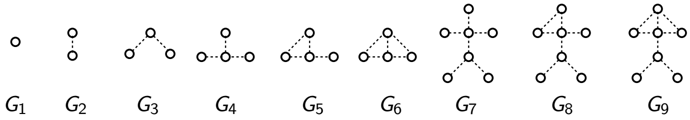
Preliminary work

- $\mathcal{G}$: Set of explorative graphs :

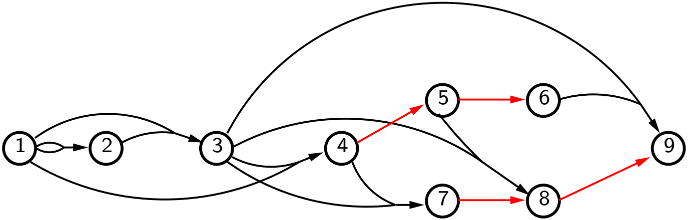


Preliminary work

- $\mathcal{G}$: Set of explorative graphs :



- Assembling graph of the final target:



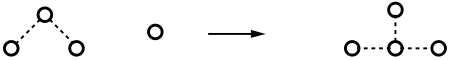
The syntax

Syntactic representation of graphs :

- Nodes = agents
- Edges = private names sharing

becomes $\langle x \rangle \ , \ \langle x, y \rangle \ , \ \langle y \rangle$

Construction rules :



becomes

$$\langle x \rangle \ , \ \langle x, y \rangle \ , \ \langle y \rangle \ , \ \langle \rangle \longrightarrow (\nu z) \left(\langle x \rangle \ , \ \langle x, y, z \rangle \ , \ \langle y \rangle \ , \ \langle z \rangle \right)$$

Formalisation of the problem

- Extraction of a core language:
  $\langle x \rangle \ , \ \langle x \rangle \ , \ \langle \rangle \ \nrightarrow \ (\nu y)(\langle x \rangle \ , \ \langle x, y \rangle \ , \ \langle y \rangle)$
  $\Longrightarrow$ restriction on synchronisation ability
- Expected property: equivalent behaviour

Formalisation of the problem

- Extraction of a core language:
  $\langle x \rangle \ , \ \langle x \rangle \ , \ \langle \rangle \ \nrightarrow \ (\nu y)(\langle x \rangle \ , \ \langle x, y \rangle \ , \ \langle y \rangle)$
  $\Longrightarrow$ restriction on synchronisation ability
- Expected property: equivalent behaviour

What does that mean ?

- Comparison of transitions
- Comparison of states

Formalisation of the problem

- Extraction of a core language:
  $\langle x \rangle \ , \ \langle x \rangle \ , \ \langle \rangle \ \nrightarrow \ (\nu y)(\langle x \rangle \ , \ \langle x, y \rangle \ , \ \langle y \rangle)$
  $\Longrightarrow$ restriction on synchronisation ability
- Expected property: equivalent behaviour

What does that mean ?

- Comparison of transitions
- Comparison of states

$\Longrightarrow$ Mathematical tool: bisimulation

Bisimulation

[Observation]
The least relation $\downarrow$ such that $S \downarrow C$ only if the connected component C appears in the system S

[(weak) bisimulation]
Binary symmetric relation $\mathfrak{R}$ such that if $S \; \mathfrak{R} \; S'$ then

1. If $S \rightarrow T$ then $\exists \; T'$ s.t. $S' \rightarrow^* T'$ and $T \; \mathfrak{R} \; T'$
2. If $S \downarrow C$ then $S' \downarrow C$.

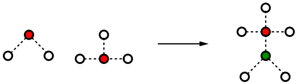
Intuitive features of the algorithm

- Only one active agent by component.
- Local knowledge of the component's structure.
- Each agent knows its role in the component.
- Propagation of the changes related to an interaction by the use of a spanning tree.

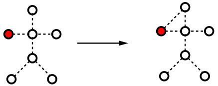
Traduction of the reactions

Set of reactions :

- Connection between 2 disjoint complexes



- Cyclic connection



- Propagation updates
- Activity switch
- Mechanism to handle the deadlocks

Connection rule

$\forall G_1, r_1, G_2, r_2$ such that $[G_1.r_1 \oplus G_2.r_2]^{abs} \in \mathcal{G}$

$$\begin{array}{c}
\langle\ S_1\ ,\ C_1\ ,\ g_1\ ,\ r_1\ ,\ Act(G_1)\ \rangle \\
\langle\ S_2\ ,\ C_2\ ,\ g_2\ ,\ r_2\ ,\ Act(G_2)\ \rangle \\
\downarrow \\
(\nu\ com) \\
\langle\ S_1 \cup \{com\}\ ,\ C_1\ ,\ g_1\ ,\ r_1\ ,\ Act(G_1.r_1 \oplus G_2.r_2)\ \rangle \\
\langle\ S_2 \cup \{com\}\ ,\ C_2\ ,\ g_1\ ,\ r_2 + \|G_1\|\ ,\ Up(S_2, \|G_1\|)\ \rangle
\end{array}$$

Self connection

$\forall G, r_1, r_2$ such that $[r_1 \overset{G}{\frown} r_2]^{abs} \in \mathcal{G}$

$$\begin{array}{c} \langle\ S_1\ ,\ C_1\ ,\ g\ ,\ r_1\ ,\ Act(G)\ \rangle \\ \langle\ S_2\ ,\ C_2\ ,\ g\ ,\ r_2\ ,\ P\ \rangle \\ \downarrow \\ (\nu\ com) \\ \langle\ S_1\ ,\ C_1 \cup \{com\}\ ,\ g\ ,\ r_1\ ,\ Act(r_1 \overset{G}{\frown} r_2)\ \rangle \\ \langle\ S_2\ ,\ C_2 \cup \{com\}\ ,\ g\ ,\ r_2\ ,\ P\ \rangle \end{array}$$

Updates

$$\begin{array}{l} \langle\ S_1\ ,\ C_1\ ,\ g_1\ ,\ r_1\ ,\ Up(L+\{com\}, Num)\ \rangle \\ \langle\ S_2+\{com\}\ ,\ C_2\ ,\ g_2\ ,\ r_2\ ,\ P\ \rangle \\ \downarrow \\ \langle\ S_1\ ,\ C_1\ ,\ g_1\ ,\ r_1\ ,\ Up(L, Num)\ \rangle \\ \langle\ S_2+\{com\}\ ,\ C_2\ ,\ g_1\ ,\ r_2+Num\ ,\ Up(S_2, Num)\ \rangle \end{array}$$

End of updates

$$\langle S, C, g, r, Up(\emptyset, Num)\rangle \longrightarrow \langle S, C, g, r, P\rangle$$

Activity switch

$$\begin{array}{c}\langle\ S_1\ ,\ C_1\ ,\ g\ ,\ r_1\ ,\ Act(G)\ \rangle\\ \langle\ S_2\ ,\ C_2\ ,\ g\ ,\ r_2\ ,\ P\ \rangle\\ \downarrow\\ \langle\ S_1\ ,\ C_1\ ,\ g\ ,\ r_1\ ,\ P\ \rangle\\ \langle\ S_2\ ,\ C_2\ ,\ g\ ,\ r_2\ ,\ Act(G)\ \rangle\end{array}$$

A more intuitive presention

Conclusions

- Formal language to express the problem of self assembling parts.
- Mathematical tool to resolve it.
- Similar work in $\mathbb{R}^n$ (to be done as in the demo).
- Extension to other fields

Conclusions

- Formal language to express the problem of self assembling parts.
- Mathematical tool to resolve it.
- Similar work in $\mathbb{R}^n$ (to be done as in the demo).
- Extension to other fields

$$\langle \quad x \quad \rangle, \qquad \langle \quad x \quad \rangle$$

Conclusions

- Formal language to express the problem of self assembling parts.
- Mathematical tool to resolve it.
- Similar work in $\mathbb{R}^n$ (to be done as in the demo).
- Extension to other fields

CAP $\langle \quad x \quad \rangle$, **AMPc** $\langle \quad x \quad \rangle$

Conclusions

- Formal language to express the problem of self assembling parts.
- Mathematical tool to resolve it.
- Similar work in $\mathbb{R}^n$ (to be done as in the demo).
- Extension to other fields

CAP $\langle \mathrm{amp}^{\mathrm{x}} \rangle$, **AMPc** $\langle \mathrm{cap}^{\mathrm{x}} \rangle$