

*Static
Analysis of
Concurrent
Programs:*

*A
Geometric
Approach,
and Perspectives*

*Eric
Goubault*

*Static
analysis*

*Geometric
semantics
and components*

*Computing
components*

ALCOOL

*Open
issues*

Static Analysis of Concurrent Programs: A Geometric Approach, and Perspectives

Eric Goubault

MSRI Computational Applications of Algebraic Topology

Modelisation and Analysis of Systems in Interaction lab

CEA/Saclay and Ecole Polytechnique

Outline of the talk

*Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*

- A very short introduction to static analysis, in view of verifying concurrent programs
- Reducing the state-space model: components categories
- “Good properties” of component categories: lifting property and van Kampen
- One simple algorithm for determining smaller state-space models
- “Constructive” lifting property
- Implementation (still simple-minded) in ALCOOL; demo and some results
- Many open questions from this work
 - Algorithmic and implementation issues
 - Mathematical issues

Typical static analysis

Static
Analysis of
Concurrent
Programs:

A

Geometric
Approach,
and Perspectives

Eric
Goubault

Static
analysis

Geometric
semantics
and components

Computing
components

ALCOOL

Open
issues

Given a program, how to prove some form of correctness?
(executions won't do)

Generally: to compute **invariants**, i.e. properties which hold true on any execution

Classical steps:

- (1) interleaving semantics, collecting semantics
- (2) abstraction (Galois connections etc.)
- (3) computation of least fixpoints - resolution of large systems of equations

We are looking for a more efficient step (1) ((2) and (3) for some other talk!)

Use of invariants

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

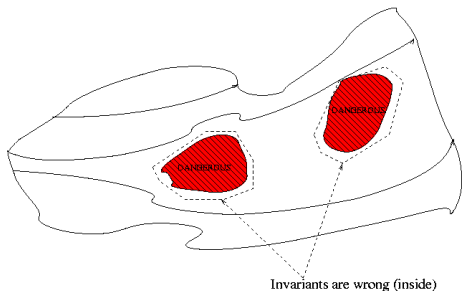
Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

From (3) we conclude in general about the “safety” (for instance absence of run-time errors) of programs:



Invariants are “integral forms” of the dynamics typically (if it were dynamical systems’ theory)

- Given a semantics by a transition system $\mathcal{T} = (S, i, E, Tran)$ (S control points, $i \in S$ initial control point, E labels, $Tran \subseteq S \times E \times S$) - essentially a labelled graph structure
- We associate to each control point s_i of \mathcal{T} an “equation” in associated variables S_i , which are sets of values that variables of the program can take, at control point s_i :

$$S_i = \bigcup_{s_j \in S \mid (s_j, t, s_i) \in Tran} \llbracket t \rrbracket S_j$$

where $\llbracket t \rrbracket$ is the interpretation of transition t , seen as a function from sets of values to sets of values

- S_i are called **invariants** at control point s_i (because it is a property of the values of variables at s_i which will be always true for any possible execution)

Example: a transition system, interpreted in sets

Static
Analysis of
Concurrent
Programs:

A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

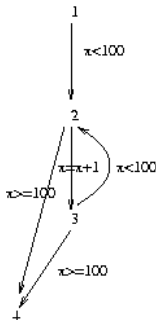
Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

```
void main() {
  int x=0; // 1
  while (x<100)
  { // 2
    x=x+1; // 3
  } // 4
}
```



$$\begin{aligned}
 x_1 &= [0, 0] \\
 x_2 &=] - \infty, 99] \cap (x_1 \cup x_3) \\
 x_3 &= x_2 + [1, 1] \\
 x_4 &= [100, +\infty[\cap (x_1 \cup x_3)
 \end{aligned}$$

$(\wp(\mathbb{N}), \subseteq)$ is a complete lattice + functional is monotonic \Rightarrow
there exists a least fixpoint (Tarsky)

(2) Now, abstractions

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

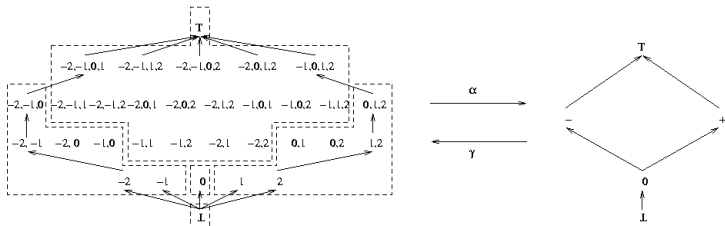
Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

- **Safe approximation of semantics:** fixpoint transfer theorems etc. through Galois connections (abstract interpretation - P. Cousot and R. Cousot)
- Here: “sign rule”:



We will use (in ALCOOL) and in the example intervals. It is so simple here that the equations are unchanged!

(3) Resolution of equations

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Perspectives

Eric
Goubault

Static
analysis

Geometric
semantics
and components

Computing
components

ALCOOL

Open
issues

- Given that the semantics (dynamics) is given by a **fixpoint equation**:

$$X = \begin{pmatrix} X_1 \\ \dots \\ X_n \end{pmatrix} = F \begin{pmatrix} X_1 \\ \dots \\ X_n \end{pmatrix}$$

- where F is monotonic, in value in a complete lattice, its lfp is the limit (Kleene) of $X^0 = \perp$, $X^1 = F(X^0)$, \dots , $X^{k+1} = X^k \cup F(X^k)$ (for an ordinal in general)

Invariant found at iteration 100

[Kleene/Jacobi/Gauss-Seidl]

- Giving the invariants:

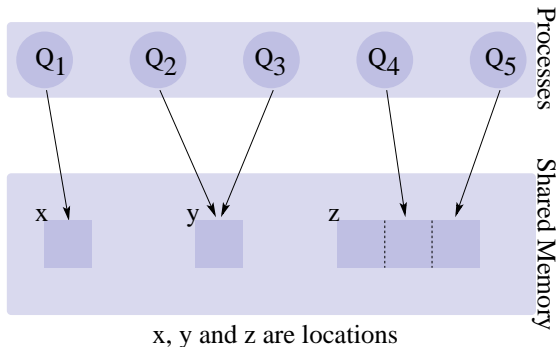
$$\begin{aligned}x_1^{100} &= [0, 0] \\x_2^{100} &=] - \infty, 99] \cap ([0, 0] \cup [1, 100]) \\ &= [0, 99] \\x_3^{100} &= [0, 99] + [1, 1] \\ &= [1, 100] \\x_4^{100} &= [100, +\infty[\cap ([0, 0] \cup [1, 100]) \\ &= [100, 100]\end{aligned}$$

- Hence no overflow in the loop, for instance. (of course, many better ways... acceleration of convergence, policy iteration, relational abstract domains etc.)

The problem in interleaving semantics is that we have **way too many states and transitions** - hence equations to solve!

Concurrent programs

shared memory style



Not sequential programs, bad states, chaotic behavior
 \implies Need for synchronizations \implies Need for locks
 \implies Interleaving semantics given by a “shuffle” of transition systems (or fibred product)

Static
 Analysis of
 Concurrent
 Programs:
 A

Geometric
 Approach,
 and Per-
 spectives

Eric
 Goubault

Static
 analysis

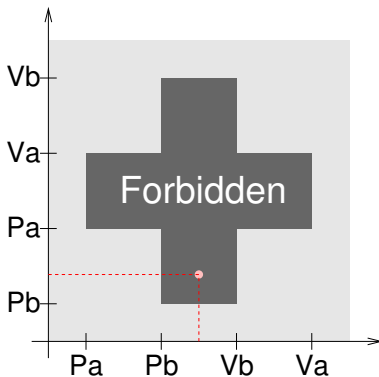
Geometric
 semantics
 and compo-
 nents

Computing
 compo-
 nents

ALCOOL

Open
 issues

$T1=Pa.Pb.Vb.Va$ in parallel with $T2=Pb.Pa.Va.Vb$



"Continuous model": $x_i =$ local time; dark grey region=**forbidden!**

see *Algebraic Topology and Concurrency* TCS 2006, L. Fajstrup, E. Goubault, M. Raussen

Execution paths are continuous

Static
Analysis of
Concurrent
Programs:

A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

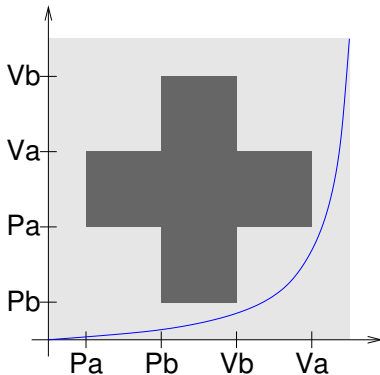
Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

$T1 = Pa.Pb.Vb.Va$ in parallel with $T2 = Pb.Pa.Va.Vb$



Traces are continuous paths increasing in each coordinate:
dipaths.

Classes of equivalent dipaths up to dihomotopy

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

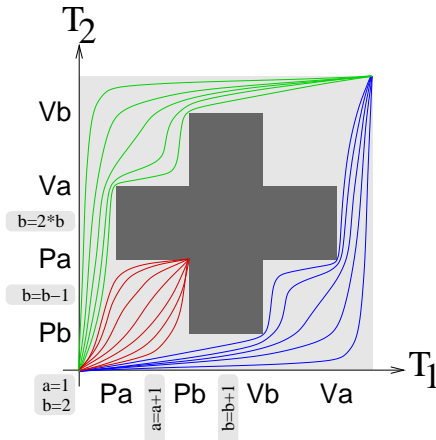
ALCOOL

Open
issues

T1 gets a and b before T2 $\Rightarrow a=2$ and $b=4$

T2 gets b and a before T1 $\Rightarrow a=2$ and $b=3$

Each of T1 and T2 gets a resource
 \Rightarrow Deadlock with $a=2$ and $b=1$



Ideally, we want to retract to...
(not quite true though)

*Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

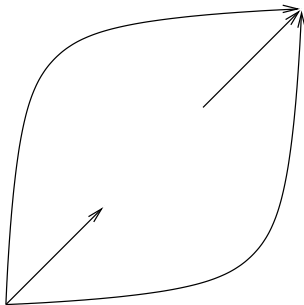
*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*



We will get back to this later.

*Static
Analysis of
Concurrent
Programs:*

A

*Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*

- **Cubical sets** (pre-existing the field of course!)
- **Po-spaces** (i.e. topological space with closed partial order), introduced first in other fields (domain theory P. Johnstone etc., fonctionnal analysis L. Nachbin etc.) **local po-spaces** (atlas of po-spaces - L. Fajstrup, E. Goubault, M. Raussen)
- **d-spaces** (M. Grandis)
- **Flows** (P. Gaucher)
- **Streams** (S. Krishnan)
- etc.

Most of the rest would apply to all of these models (except for loops!)

How to retract?

The fundamental category $\vec{\pi}_1(\vec{X})$ of a pospace \vec{X}

Static
Analysis of
Concurrent
Programs:

A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

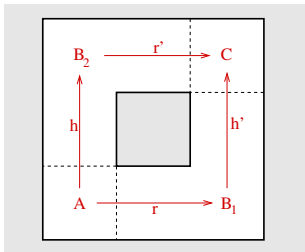
Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

- Starting with a **variation on the Poincaré groupoid**, $\pi_1(X)$ defined as the category:
 - objects: points of X ,
 - morphisms: classes of dipaths up to dihomotopy:
a morphism from x to y is a dihomotopy class $[\alpha]$ of a dipath α going from x to y .
- We see that in most interesting (to static analysis) case, it is “essentially” finite



Yoneda morphism

axiomatizing the preservation of the future and the past (1)

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

Geometric
semantics
and compo-
nents

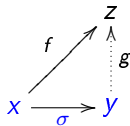
Computing
compo-
nents

ALCOOL

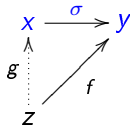
Open
issues

Let \mathcal{C} be a small category. A *Yoneda* morphism σ is an element of $\mathcal{C}[x, y]$ such that for all object z of \mathcal{C} ,

future if $\mathcal{C}[y, z] \neq \emptyset$ then for all $f \in \mathcal{C}[x, z]$, there is a unique $g \in \mathcal{C}[y, z]$ such that



past if $\mathcal{C}[z, x] \neq \emptyset$ then for all $f \in \mathcal{C}[z, y]$, there is a unique $g \in \mathcal{C}[z, x]$ such that



Yoneda system of a small category \mathcal{C}

axiomatizing the preservation of the future and the past (2)

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

Geometric
semantics
and compo-
nents

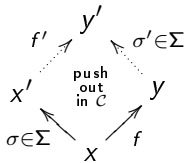
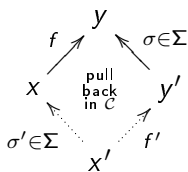
Computing
compo-
nents

ALCOOL

Open
issues

A collection Σ of morphisms of \mathcal{C} such that:

- 1 Σ is stable under composition,
- 2 Σ contains all the isomorphisms of \mathcal{C} ,
- 3 all the elements of Σ are *Yoneda* morphisms and
- 4 Σ is stable under **change** and **cochange** of base.



Define now the **component category** to be $\overrightarrow{\pi_0}(X)$ equal to the category of fractions of $\pi_1(X)$ by the maximal Yoneda System (equivalently as we shall see, as the quotient category of the same two categories).

Examples

of morphisms which do not belong to a Yoneda system

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

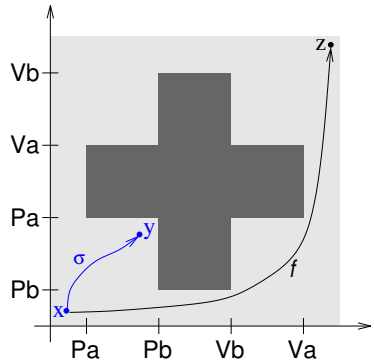
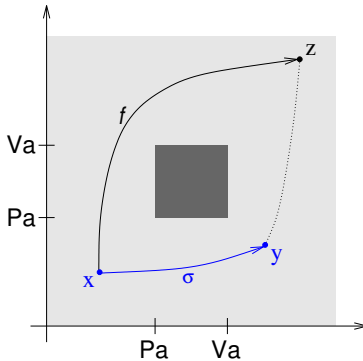
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



The category of components of the Swiss flag

Static
Analysis of
Concurrent
Programs:

A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

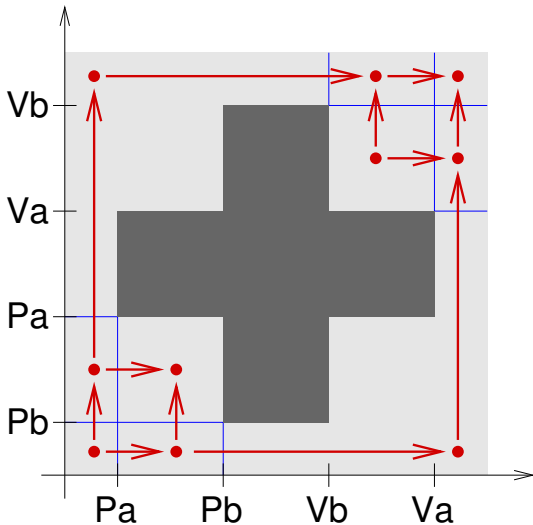
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



(the two red squares are commutative!)

The components category of the 3 philosophers

*Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

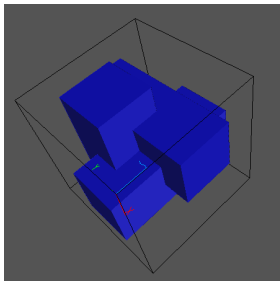
*Static
analysis*

*Geometric
semantics
and compo-
nents*

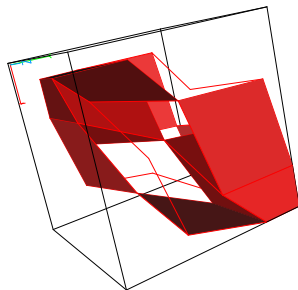
*Computing
compo-
nents*

ALCOOL

*Open
issues*



the pospace



its category of components

The components category of a 2-semaphore

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

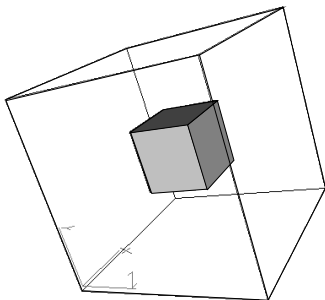
Static
analysis

Geometric
semantics
and compo-
nents

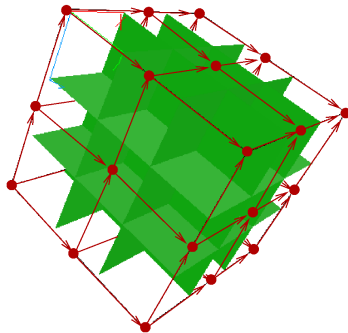
Computing
compo-
nents

ALCOOL

Open
issues



the pospace



its category of components

Notice: a certain amount of the classical π_2 is apparent; and $\overrightarrow{\pi_1}$ has no “cancellation” property in general

Lifting properties of the component category

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

Let \mathcal{C} be a category in which all endomorphisms are identities. Then there is a maximal Yoneda-system Σ in \mathcal{C} . Furthermore, let $C_1, C_2 \subset \text{Ob}(\mathcal{C})$ denote two components such that the set of morphisms (in \mathcal{C}/Σ) is *finite*. Then, for every $x_1 \in C_1$ there exists $x_2 \in C_2$ such that the **quotient map**

$$\begin{array}{ccc} \mathcal{C}(x_1, x_2) & \rightarrow & \mathcal{C}/\Sigma(C_1, C_2) \\ f & \mapsto & [f] \end{array}$$

is **bijective**.

Getting back to the Swiss flag

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Perspectives

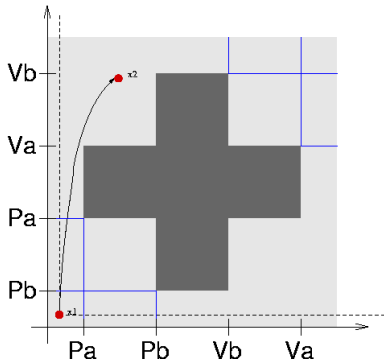
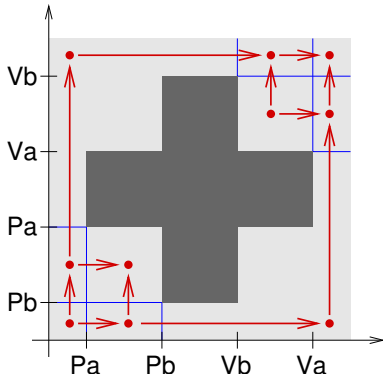
Eric
Goubault

Static
analysis
Geometric
semantics
and components

Computing
components

ALCOOL

Open
issues



Fundamental results

fractions vs quotients

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

Let \mathcal{C} be a small loop-free category and Σ a *Yoneda* system of \mathcal{C} :

- 1 the collection Σ is pure in \mathcal{C} ,
- 2 the small category \mathcal{C}/Σ is loop-free,
- 3 the small categories $\mathcal{C}[\Sigma^{-1}]$ and \mathcal{C}/Σ are equivalent and
- 4 the category $\mathcal{C}[\Sigma^{-1}]$ is fibered over \mathcal{C}/Σ .
- 5 Seifert/van Kampen on component categories
- 6 if \vec{K} is a compact pospace, then any component of $\vec{\pi}_1(\vec{K})$ has both a **greatest lower bound** and an **least upper bound** in $(|K|, \sqsubseteq)$.

see *Ph. D. Thesis* of E. Haucourt

see *Components of the Fundamental Category* - APCS 04, L. Fajstrup, E.

Goubault, E. Haucourt, M. Raussen

see also *Components of the Fundamental Category II* - APCS 07, E. Goubault, E.

Haucourt

Interest of components for static analysis

*Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*

- New “control points” or “states”: objects of the component category (much more economical than interleaving - see figures at the end of the talk)
- New “transitions”: morphisms of the component category
- So we should see **how to compute components on spaces coming from the semantics of concurrent programs?**
- And we should see how to give a **constructive version of the “lifting property”** (to be able to compute $\llbracket \cdot \rrbracket$ on the newly defined transitions!)

Geometric interpretation of the components - duality



Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

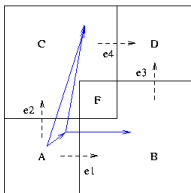
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



The components A , B , C and D correspond to the squares separated by the **horizontal and vertical lines** from the **min and max points** of the forbidden region F .

- “**Duality**”: we identify e_1 with the codimension 1 linear variety (here, the vertical segment, orthogonal to e_1)
- Similarly, e_2 is identified with the horizontal line left of the min point of F etc.
- There is no interesting **codimension 2** linear variety here, hence no relation between morphisms.

Example

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Perspectives

Eric
Goubault

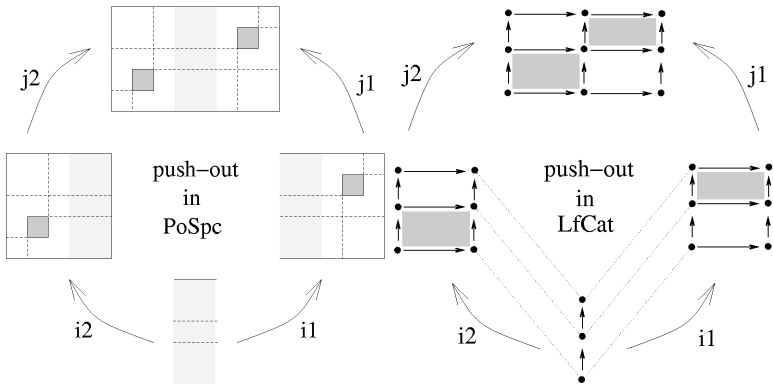
Static
analysis

Geometric
semantics
and components

Computing
components

ALCOOL

Open
issues



Geometric interpretation of the induction step: “duality”



Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Perspectives

Eric
Goubault

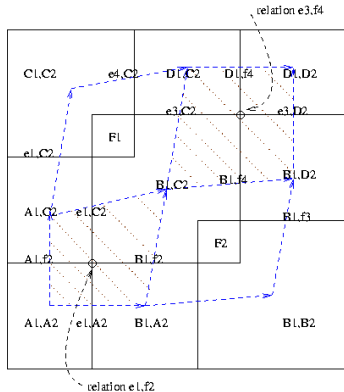
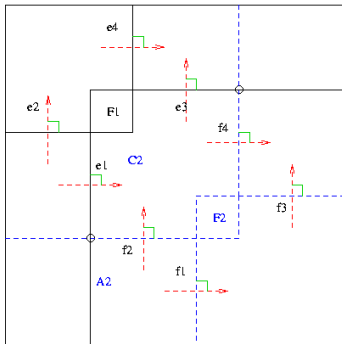
Static
analysis

Geometric
semantics
and components

Computing
components

ALCOOL

Open
issues



(proof in “generic” situations by van Kampen)

Inductive presentation of the component category

- CONCUR'05, E. Goubault & E. Haucourt



Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

- Component categories (in the classical concurrency theory setting) are generated by **2-dimensional pre-cubical sets**
- Base case is OK, now the induction step: given the component category of $[0, 1]^n \setminus R$ generated by a 2-dimensional precubical set $(Y_0, Y_1, Y_2, \delta^0, \delta^1)$, define a new structure $(Z_0, Z_1, Z_2, \partial^0, \partial^1)$ which will generate (an “approximation” of) the component category of $U \setminus R$:
 - $Z_0 = \{A \cap B \mid A \in X_0, B \in Y_0, A \cap B \neq \emptyset\}$
 - $Z_1 = \begin{aligned} &\{A \cap f \mid A \in X_0, B \in Y_1, A \cap f \neq \emptyset\} \\ &\cup \{e \cap B \mid e \in X_1, B \in Y_0, e \cap B \neq \emptyset\} \end{aligned}$
 - $Z_2 = \begin{aligned} &\{e \cap f \text{ “non degenerate”} \mid e \in X_1, f \in Y_1, e \cap f \neq \emptyset\} \\ &\cup \{R \cap B \mid R \in X_2, B \in Y_0, R \cap B \neq \emptyset\} \\ &\cup \{A \cap S \mid A \in X_0, S \in Y_2, A \cap S \neq \emptyset\} \end{aligned}$

Some figures - component computations

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

(with the current naive implementation)

n	sec	Mb	# o	# m	# r	# p	#s	# t
3	0.38	≤ 10	27	48	18	6	576	1475
4	0.43	≤ 15	85	200	132	24	3966	13450
5	0.69	19	263	770	730	120	27265	113938
6	3.49	23	807	2832	3516	720	184876	914019
7	96	42	2467	10094	15484	5040	?	?
8	1656	100	7533	35216	64312	40320	?	?
9	13739	319	22995	120924	256158	362880	2996970	22698700

Might be an overapproximation

Static
Analysis of
Concurrent
Programs:

A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

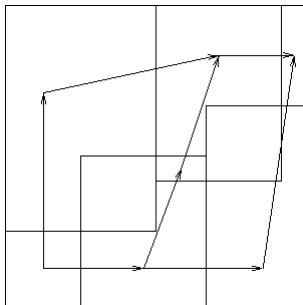
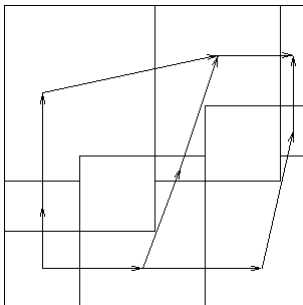
Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

I.e. not the maximal Yoneda system



Principle of the static analyser ALCOOL

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Perspectives

Eric
Goubault

Static
analysis

Geometric
semantics
and components

Computing
components

ALCOOL

Open
issues

- Continuous semantics (**deformable**)
 - somehow, for lack of a better theory on the combinatorial side [**cubical sets**]
- Category of paths modulo deformation (“**components**”)
- **Essential** continuous paths
 - i.e. **one path per equivalence class**
- Sequential traces corresponding to these paths
 - **lifting property**
- Classical sequential analysis
 - **abstract interpretation based static analysis** (“collecting” only on the essential paths): here using intervals

Based on an intermediary language (sort of a process algebra with values, simple expressions, tests and while loops, semaphores, FIFO - bounded and unbounded - queues, monitors and synchronisation barriers)

Example program, in ALCOOL's language

*Static
Analysis of
Concurrent
Programs:*

A

*Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*

```
#fifo x  
#fifo y  
#sem z  
#sem evt
```

```
INIT=@(a,5).@(z,0).@(evt,[0,2])  
PROG=automate|tache
```

```
act1=R(x,z).@(z,z*2)  
act2=R(y,z).@(z,z*3+1)  
act3=Pa.@(a,1).Va
```

*Static
Analysis of
Concurrent
Programs:*

*A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*

```
ligneA=act1+[a=0] - (act2+[a=1] - (act3+[a=2] -))  
ligneB=act2+[a=0] - (act3+[a=1] - (act1+[a=2] -))  
ligneC=act3+[a=0] - (act1+[a=1] - (act2+[a=2] -))
```

```
matrice=ligneA+[evt=0] - (ligneB+[evt=1] -  
                           (ligneC+[evt=2] -))
```

```
automate=matrice.automate
```

```
tache=S(x,7).S(y,9).Pa.@(a,0).Va.Pa.@(a,2).Va
```

Example of other concurrency constructs

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

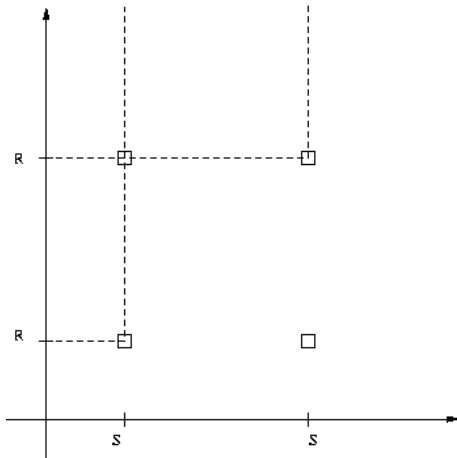
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



unbounded FLEO

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Perspectives

Eric
Goubault

Static
analysis

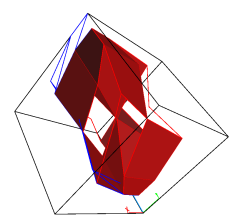
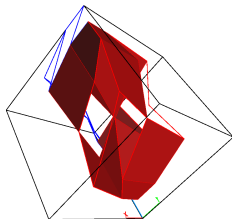
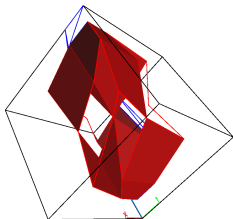
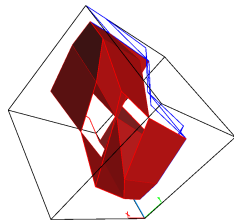
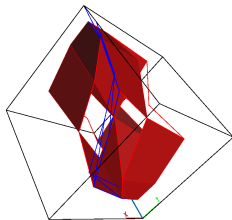
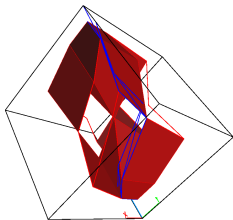
Geometric
semantics
and components

Computing
components

ALCOOL

Open
issues

3 philosophers: 6 (=3!) maximal legal paths



Syntactic representation of a path...

...or how to choose a “good” lift:

- From a continuous path, we want to get back to a “discrete” path
- This “discrete” path should be an **interleaving** path corresponding to this idealized execution
- This can then be analyzed by any standard **sequential analyzer**

For this, we remark that:

- (1) Every component has a **trivial** $\vec{\pi}_1$
- (2) There exists a path (unique) from the **minimum** (or infimum in general) from a component to the minimum of the next component (essentially by the lifting property) (modulo some technicalities I am hiding here...)

Given the morphisms of the component category, we compute:

- (a) the minimum of the components (i.e. of hyperrectangles minus the forbidden region)
- (b) the **program** comprising the possible executions between the minimum of a component, and the minimum of the next
- (c) we use the interleaving semantics for finding *1* path in this program (in a very economical manner)

Discretization of paths

*Static
Analysis of
Concurrent
Programs:*

*A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

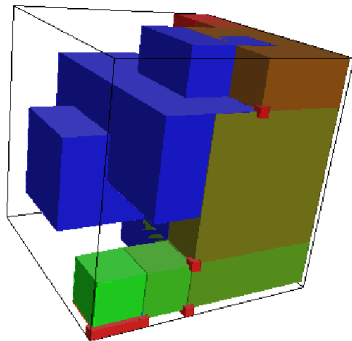
*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*



0 | 0 | P(c)

In context #sem c 1, #sem b 1, and #sem a 1.

Discretization of paths

*Static
Analysis of
Concurrent
Programs:*

*A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

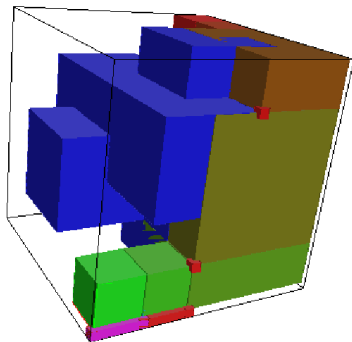
*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*



0 | 0 | P(a)

In context #sem c 0, #sem b 1, #sem a 1.

Discretization of paths

*Static
Analysis of
Concurrent
Programs:*

*A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

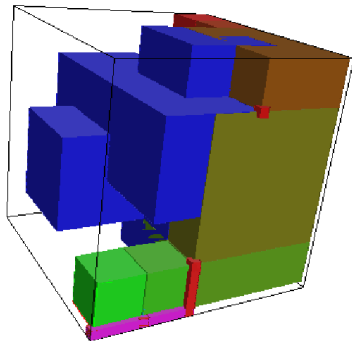
*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*



0 | P(b) | 0

In context #sem c 0, #sem b 1, #sem a 0.

Discretization of paths

Static
Analysis of
Concurrent
Programs:

A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

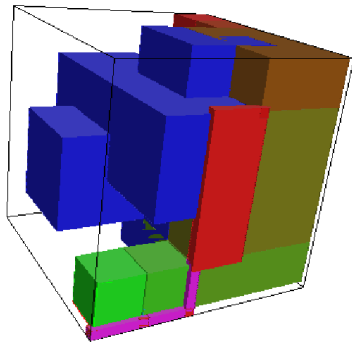
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



0 | P(c).V(b).V(c) | V(c)

In context #sem c 0, #sem b 0, #sem a 0.

Discretization of paths

Static
Analysis of
Concurrent
Programs:

A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

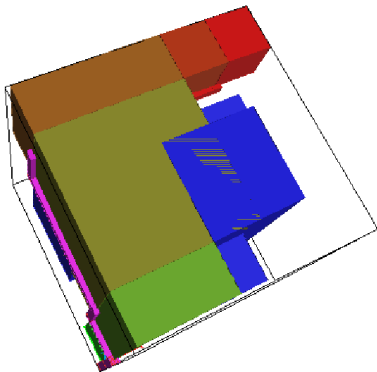
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

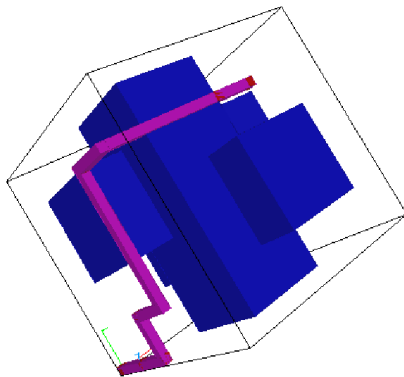
Open
issues



$V(b) \mid 0 \mid 0$

In context #sem c 1, #sem b 0, #sem a 1.

Discretisation of paths



$\{3\}P(c); \{3\}P(a); \{2\}P(b); \{3\}V(c); \{2\}P(c); \{2\}V(b);$
 $\{2\}V(c); \{3\}V(a); \{1\}P(a); \{1\}P(b); \{1\}V(a); \{1\}V(b);$

*Static
 Analysis of
 Concurrent
 Programs:*

*A
 Geometric
 Approach,
 and Per-
 spectives*

*Eric
 Goubault*

*Static
 analysis*

*Geometric
 semantics
 and compo-
 nents*

*Computing
 compo-
 nents*

ALCOOL

*Open
 issues*

Discretisation of paths (2 - deadlock)

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

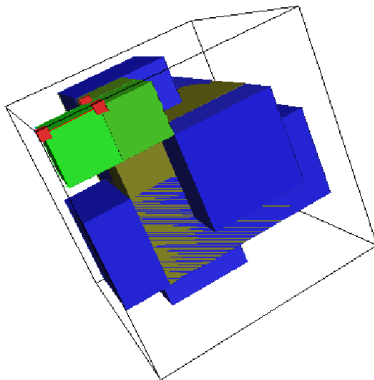
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



$\{1\}P(a); \{3\}P(c); \{2\}P(b);$

Discretisation of paths (3)

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

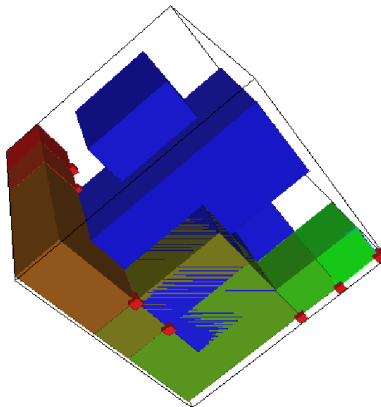
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



$\{2\}P(b); \{2\}P(c); \{2\}V(b); \{2\}V(c); \{3\}P(c); \{3\}P(a);$
 $\{3\}V(c); \{3\}V(a); \{1\}P(a); \{1\}P(b); \{1\}V(a); \{1\}V(b);$

Discretisation of paths (4)

Static
Analysis of
Concurrent
Programs:

A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

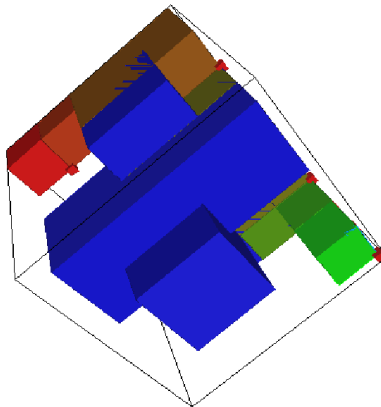
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



$$\{1\}P(a); \{1\}P(b); \{3\}P(c); \{1\}V(a); \{3\}P(a); \{3\}V(c);$$

$$\{3\}V(a); \{1\}V(b); \{2\}P(b); \{2\}P(c); \{2\}V(b); \{2\}V(c);$$

Discretisation of paths (5)

Static
Analysis of
Concurrent
Programs:

A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

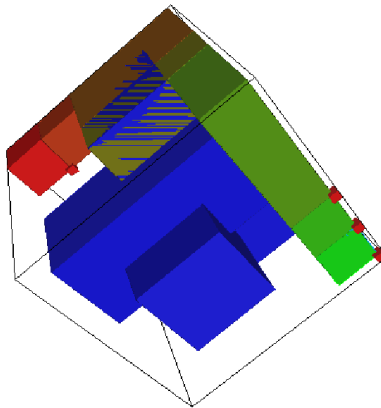
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



$$\{3\}P(c); \{3\}P(a); \{3\}V(c); \{3\}V(a); \{1\}P(a); \{1\}P(b);$$

$$\{1\}V(a); \{1\}V(b); \{2\}P(b); \{2\}P(c); \{2\}V(b); \{2\}V(c);$$

Discretisation of paths (6)

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

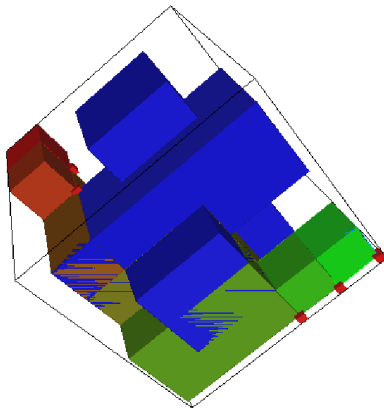
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



$\{2\}P(b); \{2\}P(c); \{1\}P(a); \{2\}V(b); \{1\}P(b); \{1\}V(a);$
 $\{2\}V(c); \{3\}P(c); \{3\}P(a); \{3\}V(c); \{3\}V(a); \{1\}V(b);$

Discretisation of paths (γ)

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

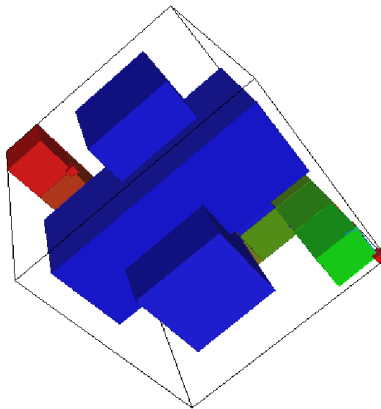
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues



$$\{1\}P(a); \{1\}P(b); \{1\}V(a); \{1\}V(b); \{2\}P(b); \{2\}P(c);$$

$$\{2\}V(b); \{2\}V(c); \{3\}P(c); \{3\}P(a); \{3\}V(c); \{3\}V(a);$$

*Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Perspectives*

*Eric
Goubault*

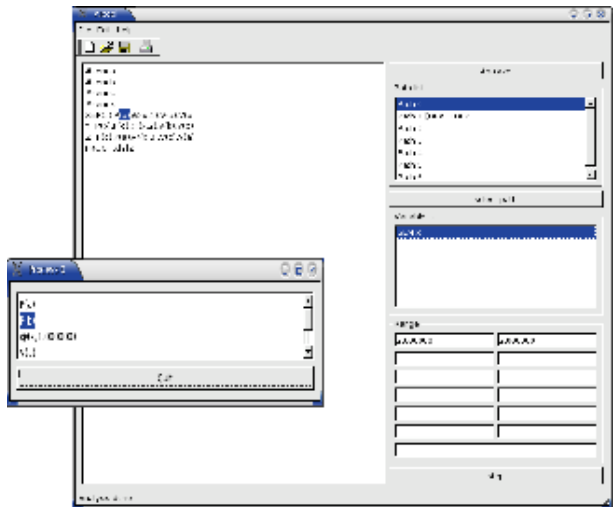
*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*



*Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*

- Finds deadlocks, unreachables and loss of messages
- Identifies concrete, essential coordination paths
- GDB like (“abstract testing”) browsing of the paths
- About 32000 lines of C code

FIRST DEMO...

And in reality...

- Code analysis (**actuation and control**)
- Application of about **100000** lines of C, a dozen threads, about 10 semaphores, message queues, external events etc.
- Deadlocks, reachability, **potential loss of messages**, local invariants etc.

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

Static
analysis

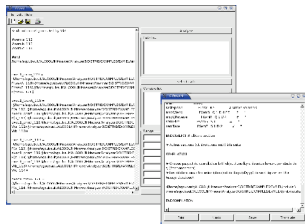
Geometric
semantics
and compo-
nents

Computing
compo-
nents

ALCOOL

Open
issues

→ **Two analysers**: MIEL (J.-M. Collart/G. Canet) - interactive modeling - ALCOOL (E. Goubault) on the produced model



First results of ALCOOL for a real code

*Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*

- sub-group of 6 “periodical” processes
- precision of the analysis at minimal: 47 Mb, 8 minutes 30s
- The analyser proves that there is no **deadlock, nor loss of message**
- (under some simplifying assumptions)

“Periodical” sub-model

Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives

Eric
Goubault

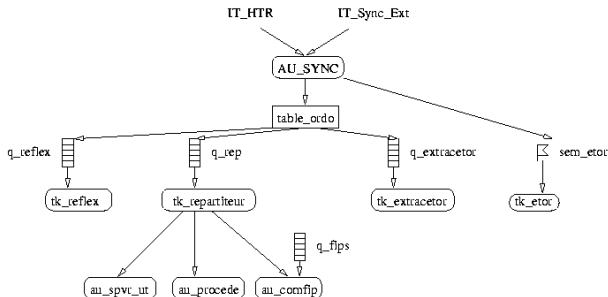
Static
analysis

Geometric
semantics
and compo-
nents

Computing
compo-
nents

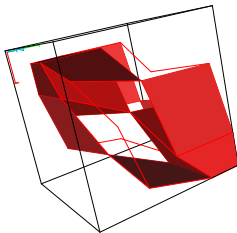
ALCOOL

Open
issues



- Computation of components:
 - still **sub-optimal** (in size)
 - use of static/dynamic segment trees to improve the computation of intersections
 - (simpler) use of geometric simple geometric constraints to improve the computation of intersections
 - etc.
- Computation of equivalence classes of paths (i.e. morphisms in the component category)
 - (simpler) approximation of relations (**persistent sets** Godefroid/Wolper)
 - Plus all classical improvements: symbolic terms, symmetry, on the fly traversal, state space hashing etc.

- Computation of morphisms in the component category:
 - Help from **homology**? (many tries!)
 - (on the longer run) simplification of the retract by considering also the $\vec{\pi}_n$ ["homotopical resolution", or "higher-order syzygies"], example:



1st order: [a,b],
[b,c], [a,c]
2nd order: [a,b,c]

- For most analysis, component in the future (or in the past) would suffice: **Yoneda₊ systems** are:
 - all σ in Σ_+ are monos in \mathcal{C} , and Σ_+ contains $Iso(\mathcal{C})$
 - Σ_+ is stable under pushout (with any morphism in \mathcal{C})
 - If there is $u : \beta \rightarrow \gamma$ in \mathcal{C} , then for all $\sigma : \alpha \rightarrow \beta$ in Σ_+ , and all $f : \alpha \rightarrow \gamma$ in \mathcal{C} , f factors through σ , that is, there exists $h : \beta \rightarrow \gamma$ such that the following diagram commutes



- How to compute the corresponding “future” components?
(**critical points** of some sort?)
 \implies See M. Raussen talk?

*Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

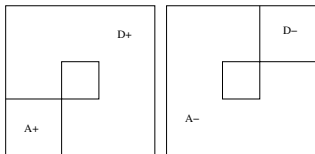
*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

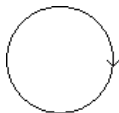
ALCOOL

*Open
issues*



- There is a **lifting** property
- Relation with **orthogonal subcategories**
- In some cases (conjectured to be those of interest for applications to concurrency theory): **reflective in $\pi_1(X)$**

- Components in the presence of **loops/non-determinism**:



- the category of components is exactly π_1 (continuum of points)!
 - difficult to find a direct notion of Yoneda system in that case
 - we would like something like components $\sim (\mathbb{N}, +)$
- Should be coherent with the view that one would have when looking at the **universal discovering** (if it exists - see L. Fajstrup's talk)
- See also M. Raussen's proposal

*Static
Analysis of
Concurrent
Programs:
A
Geometric
Approach,
and Per-
spectives*

*Eric
Goubault*

*Static
analysis*

*Geometric
semantics
and compo-
nents*

*Computing
compo-
nents*

ALCOOL

*Open
issues*

- **dihomotopy equivalence?** Algebraic homotopical structure (Quillen, Baues...)?
⇒ leads in work by M. Grandis, P. Gaucher, K. Worytkiewicz etc.
- **Classification** of models modulo dihomotopy equivalence? (at least in dimension 2?)
⇒ right “primitive” **synchronisation primitives?** (foundational work, concurrent Turing machine?)
- **Enrichment** of the model, e.g. timing issues:
⇒ geodesics, CAT0 conditions (see R. Ghrist/V. Paterson)

- Right directed notions in **cubical sets**?
⇒ would be much more natural in the context of semantics and static analysis - leads in work (classical case) of R. Jardine
 - What is the correct category involved?
 - Can we define directly an algebraic homotopical structure (for the directed equivalence)? Quillen equivalence with the (di-)topological case?
- Relation with **causality** issues in theoretical physics (general relativity)?
⇒ see Penrose models, P. Panangaden/K. Martin work

Thanks for your attention!