How to write and prove programs with constraints and linear logic?

Thierry Martinez
Contraintes Project-Team

INRIA Junior Seminar, 18 October 2011

“Contraintes” project-team

Topic: Formal semantics for programming languages

Methods: Logic and constraints

Applications:

- Solving/optimization of combinatorial problems
- Systems Biology

"Contraintes" project-team

Topic	Formal semantics for ~~programming~~ modeling languages
Methods	Logic and constraints
Applications	▸ Solving/optimization of combinatorial problems ▸ Systems Biology

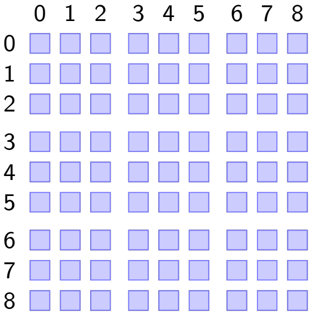
Sudoku

We probably all know the rules of the Sudoku...

	0	1	2	3	4	5	6	7	8
0	□	□	□	□	□	□	□	□	□
1	□	□	□	□	□	□	□	□	□
2	□	□	□	□	□	□	□	□	□
3	□	□	□	□	□	□	□	□	□
4	□	□	□	□	□	□	□	□	□
5	□	□	□	□	□	□	□	□	□
6	□	□	□	□	□	□	□	□	□
7	□	□	□	□	□	□	□	□	□
8	□	□	□	□	□	□	□	□	□

Sudoku

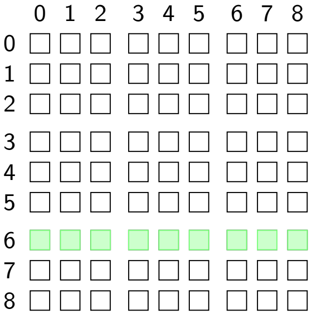
We probably all know the rules of the Sudoku...



- for every line i and every column j, the case (i, j) should have a value $1 \leqslant X_{(i,j)} \leqslant 9$.

Sudoku

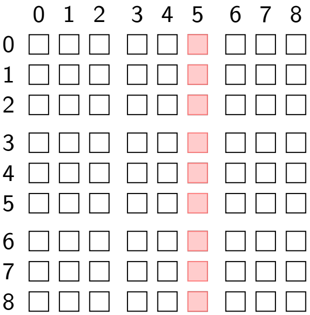
We probably all know the rules of the Sudoku...



- for every line i and every column j, the case (i, j) should have a value $1 \leqslant X_{(i,j)} \leqslant 9$.
- for every line i and every pair (j, k) of distinct columns, we should have $X_{(i,j)} \neq X_{(i,k)}$.

Sudoku

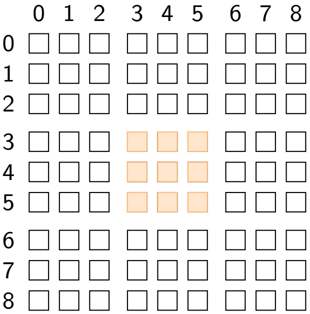
We probably all know the rules of the Sudoku...



- for every line i and every column j, the case (i, j) should have a value $1 \leqslant X_{(i,j)} \leqslant 9$.
- for every line i and every pair (j, k) of distinct columns, we should have $X_{(i,j)} \neq X_{(i,k)}$.
- for every column i and every pair (j, k) of distinct lines, we should have $X_{(j,i)} \neq X_{(k,i)}$.

Sudoku

We probably all know the rules of the Sudoku...



- for every line i and every column j, the case (i,j) should have a value $1 \leqslant X_{(i,j)} \leqslant 9$.
- for every line i and every pair (j,k) of distinct columns, we should have $X_{(i,j)} \neq X_{(i,k)}$.
- for every column i and every pair (j,k) of distinct lines, we should have $X_{(j,i)} \neq X_{(k,i)}$.
- for every 3×3-block (i,j) and every distinct cases (m,n) and (m',n') in this block, we should have $X_{3\times(i,j)+(m,n)} \neq X_{3\times(i,j)+(m',n')}$.

Sudoku

We probably all know the rules of the Sudoku...

Logical formulas

- $\forall i\, j \in \{0 \ldots 8\}, 1 \leqslant X_{(i,j)} \leqslant 9$
- $\forall i\, j\, k \in \{0 \ldots 8\}, j \neq k \Rightarrow X_{(i,j)} \neq X_{(i,k)}$
- $\forall i\, j\, k \in \{0 \ldots 8\}, j \neq k \Rightarrow X_{(j,i)} \neq X_{(k,i)}$
- $\forall i\, j\, m\, n\, m'\, n' \in \{0 \ldots 2\}, (m, n) \neq (m', n') \Rightarrow X_{3\times(i,j)+(m,n)} \neq X_{3\times(i,j)+(m',n')}$

Constraints

- Constraints = atomic formulas, $X_{(1,1)} \neq X_{(1,2)}$
- Model = conjunction of constraints

$$\bigwedge \text{constraints} \Rightarrow \text{solution}$$

- Constraints formalized as relations:

$$\text{“}X_{(1,1)} \neq X_{(1,2)}\text{”} = \{(X_{(i,j)})_{0 \leqslant i \leqslant 8, 0 \leqslant j \leqslant 8} \mid X_{(1,1)} \neq X_{(1,2)}\}$$

- The set of solutions is the intersection

$$\bigcap \{\text{relations}\} = \{\text{set of solutions}\}$$

- Explicit representation is intractable

Domain and propagation

x	2	3		5			8	9
4			7				2	3
7								
		5		9	7	2	1	
8				1				5
								8
6		1	9				3	
9	7				2			

Domain and propagation

x	2	3		5			8	9
4			7				2	3
7								
		5		9	7	2	1	
8				1				5
								8
6		1	9				3	
9	7				2			

Domain:

$x \longrightarrow$ 1 2 3 4 5 6 7 8 9

Domain and propagation

x	2	3		5			8	9
4			7				2	3
7								
		5		9	7	2	1	
8				1				5
								8
6		1	9				3	
9	7				2			

Domain:

$x \longrightarrow$	1	2	3	4	5	6	7	8	9

Domain and propagation

x	2	3		5			8	9
4			7				2	3
7								
		5		9	7	2	1	
8				1				5
								8
6		1	9				3	
9	7				2			

Domain:

$x \longrightarrow$	1	2	3	4	5	6	7	8	9

Domain and propagation

x	2	3		5			8	9
4			7				2	3
7								
		5		9	7	2	1	
8				1				5
								8
6		1	9				3	
9	7				2			

Domain:

$x \longrightarrow$ 1 2 3 4 5 6 7 8 9

Domain and propagation

1	2	3		5			8	9
4			7				2	3
7								
		5		9	7	2	1	
8				1				5
								8
6		1	9				3	
9	7				2			

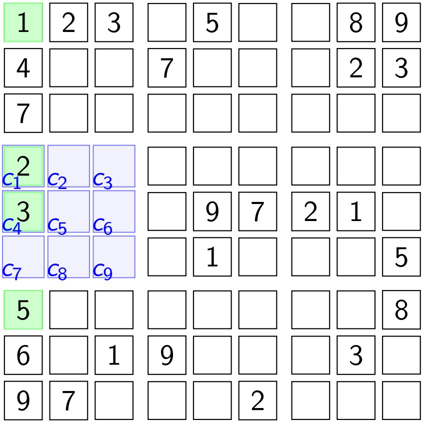
Domain:

$x \longrightarrow$ 1 2 3 4 5 6 7 8 9

Domain and propagation

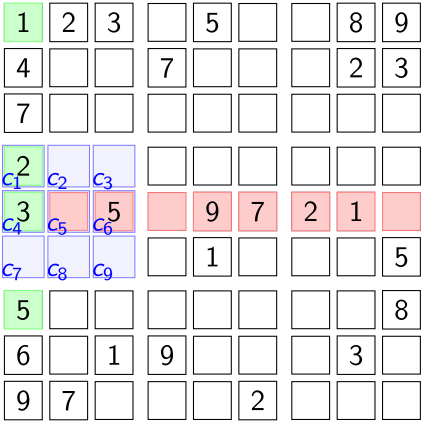
1	2	3		5			8	9
4			7				2	3
7								
2								
3		5		9	7	2	1	
8				1				5
5								8
6		1	9				3	
9	7				2			

Domain and propagation



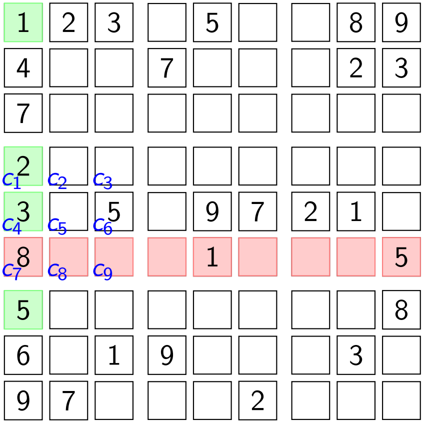
There exists $x \in 1, \dots, 9$ such that $c_x = 1$.

Domain and propagation



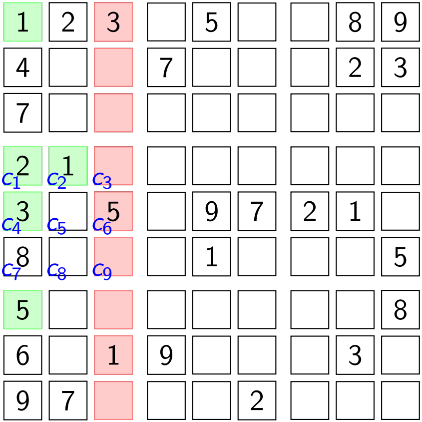
There exists $x \in 1, \dots, 9$ such that $c_x = 1$.

Domain and propagation



There exists $x \in 1, \ldots, 9$ such that $c_x = 1$.

Domain and propagation



There exists $x \in 1, \ldots, 9$ such that $c_x = 1$.

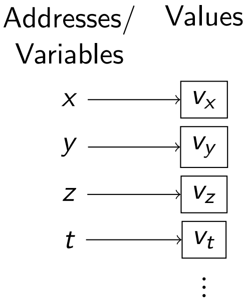
$x \longrightarrow$

Domain and propagation

1	2	3		5		7	8	9
4			7				2	3
7				2				
2	1					8		7
3		5	8	9	7	2	1	
8		7	2	1		3		5
5	3	2				9	7	8
6		1	9	7			3	2
9	7				2			

Memory paradigm shift

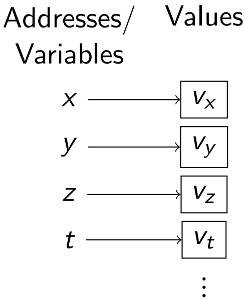
RAM model



- Imperative paradigm: assigns many, reads many
- Functional paradigm: assigns once, reads many

Memory paradigm shift

RAM model



Constraint memory model
(Partial information)

There exist x, y, z, t... such that

increasing knowledge

Memory paradigm shift

RAM model

Addresses/ Variables — Values

$x \longrightarrow \boxed{v_x}$

$y \longrightarrow \boxed{v_y}$

$z \longrightarrow \boxed{v_z}$

$t \longrightarrow \boxed{v_t}$

$\vdots$

Constraint memory model
(Partial information)

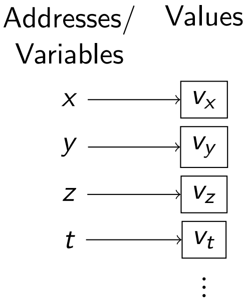
There exist x, y, z, t... such that

$x \in \{1, \ldots, 15\}$

increasing knowledge ↓

Memory paradigm shift

RAM model



Constraint memory model (Partial information)

There exist x, y, z, $t \ldots$ such that
$x \in \{1, \ldots, 15\}$ and
$y \in \{5, \ldots, 50\}$

increasing knowledge ↓

Memory paradigm shift

RAM model

Addresses/Variables — Values

$x \longrightarrow v_x$

$y \longrightarrow v_y$

$z \longrightarrow v_z$

$t \longrightarrow v_t$

$\vdots$

Constraint memory model (Partial information)

There exist x, y, z, $t \ldots$ such that

$x \in \{1, \ldots, 15\}$ and

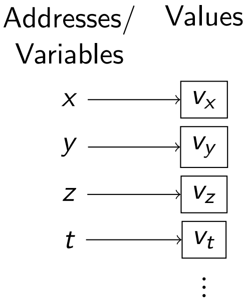
$y \in \{5, \ldots, 50\}$ and

$y \leqslant x$

increasing knowledge ↓

Memory paradigm shift

RAM model



Constraint memory model
(Partial information)

There exist x, y, z, $t \ldots$ such that
$x \in \{\not{1}\ 5, \ldots, 15\}$ and
$y \in \{5, \ldots, 50\}$ and
$y \leqslant x$

increasing knowledge ↓

Memory paradigm shift

RAM model

Addresses/ Variables		Values
x	$\longrightarrow$	v_x
y	$\longrightarrow$	v_y
z	$\longrightarrow$	v_z
t	$\longrightarrow$	v_t
		$\vdots$

Constraint memory model
(Partial information)

There exist x, y, z, $t \ldots$ such that

$x \in \{\cancel{1}\ 5, \ldots, 15\}$ and

$y \in \{5, \ldots, \cancel{50}\ 15\}$ and

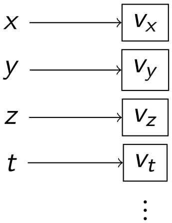
$y \leqslant x$

increasing knowledge $\downarrow$

Memory paradigm shift

RAM model

Addresses/ Variables Values

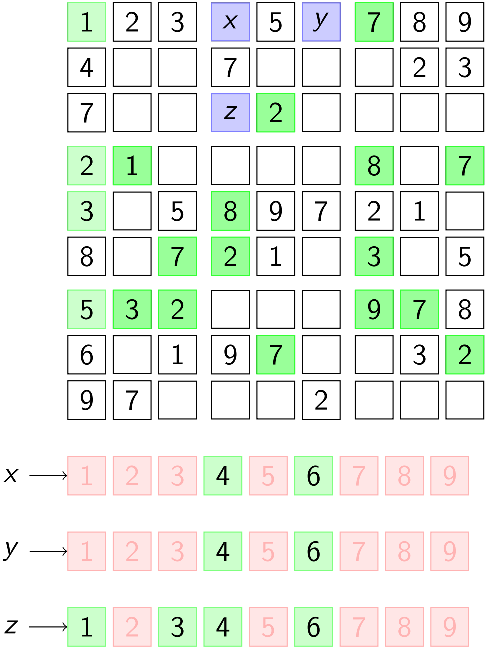


Constraint memory model
(Partial information)

There exist x, y, z, t... such that
$x \in \{\cancel{1}\ 5, \dots, 15\}$ and
$y \in \{5, \dots, \cancel{50}\ 15\}$ and
$y \leqslant x$ and
$z \in \mathbb{Q} \cap [5, 9]$
and more...

increasing knowledge ↓

Propagation power



Propagation power

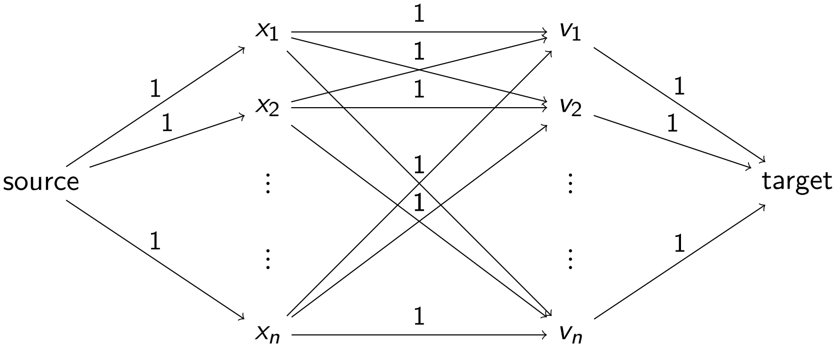
1	2	3	x	5	y	7	8	9
4			7				2	3
7			z	2				
2	1					8		7
3		5	8	9	7	2	1	
8		7	2	1		3		5
5	3	2				9	7	8
6		1	9	7			3	2
9	7				2			

$x \longrightarrow$	1	2	3	4	5	6	7	8	9
$y \longrightarrow$	1	2	3	4	5	6	7	8	9
$z \longrightarrow$	1	2	3	4	5	6	7	8	9

Propagation power

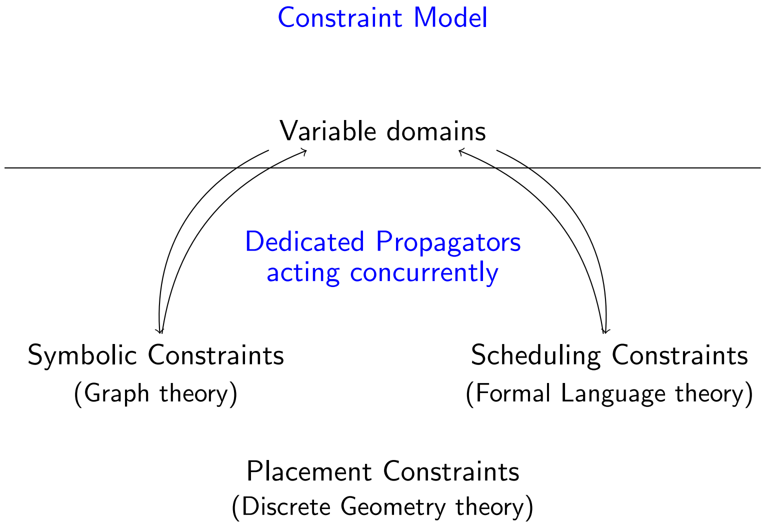
1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	1	4	3	6	5	8	9	7
3	6	5	8	9	7	2	1	4
8	9	7	2	1	4	3	6	5
5	3	2	6	4	1	9	7	8
6	4	1	9	7	8	5	3	2
9	7	8	5	3	2	6	4	1

Flow-network algorithm



Residual network of Ford-Fulkerson: reduced domain

Concurrent programming framework

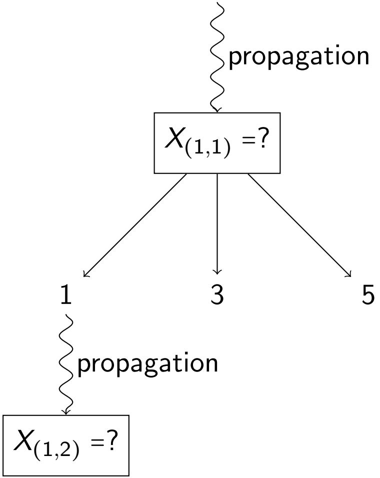


NP-completeness

1	2	3			6	7		
		6	7		9	1	2	3
			1	2	3		5	6
2	1		3	6	5	8		7
3	6		8	9	7	2	1	
8			2	1	4	3	6	
	3	1	6		2			
6			9	7	1		3	2
9		2		3	8	6		1

Propagators are polynomial. Finding a solution is NP-complete.

Propagation and search



Andorra Principle

Do the deterministic bits first.

Conjunction and disjunction

- In constraint programming, "and" between constraint
- "or" to express choices: in Sudoku,
  $X_{(1,1)} = 1 \vee X_{(1,1)} = 2 \vee \cdots \vee X_{(1,1)} = 9$

Logic programming: logic as a programming language

- Abstracting programming traits: concurrency, non determinism...

- Every computation is the search for a proof

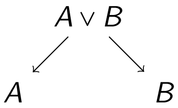
Programs = Logical formulas
Execution = Proof search

What is a proof for a conjunction?

$$\dfrac{\dfrac{\vdots}{A} \qquad \dfrac{\vdots}{B}}{A \land B}$$

What is a proof for a disjunction?

$$\frac{\dfrac{\vdots}{A}}{A \vee B} \qquad\qquad \frac{\dfrac{\vdots}{B}}{A \vee B}$$



The logical implication as synchronization mechanism

$$\dfrac{\dfrac{\vdots}{A} \qquad \dfrac{\vdots}{A \Rightarrow B}}{B}$$

Logic operators as programming constructs

- "and", $\wedge$: parallel composition
- "or", $\vee$: non-deterministic choice
- "implies", $\Rightarrow$: synchronization between parallel tasks (wait)
- "exists", $\exists$: introducing local variables
- elementary formulas: constraints, for adding knowledge about variables

To implement propagators, need to update domains (imperative features).

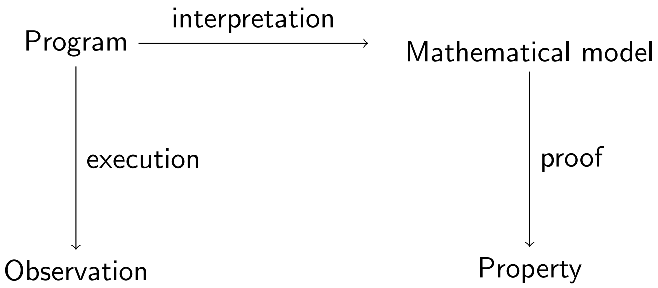
The Linear Concurrent Constraint Programming project

- Linear logic (Girard, 87): logic where formulas are resources
- Linear implication $A \multimap B$ is a process which transforms and consumes A to produce B
- Synchronization mechanism relying on linear implication updates the knowledge by removing some hypotheses

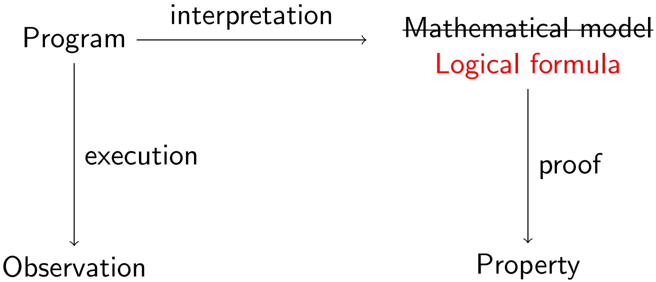
Linear logic as a concurrent programming language

- Constraints = messages, with partial knowledge
- Logic variables = communication channel
- Existential operator ($\exists$) = channel locality
- Universal operator ($\forall$) = generic synchronization
  $(\forall x(a(x) \multimap \dots))$

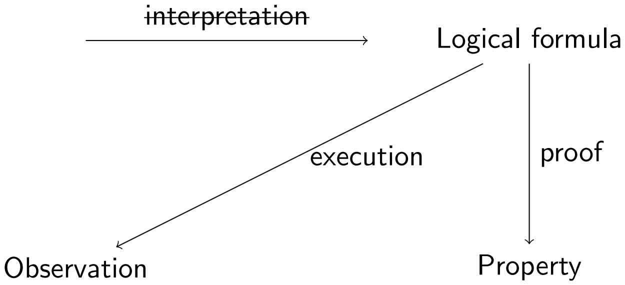
Semantics of programming languages



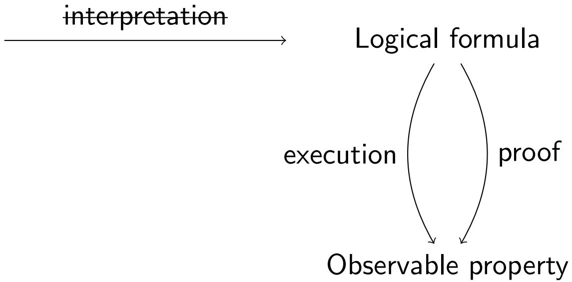
Semantics of programming languages



Semantics of programming languages



Semantics of programming languages



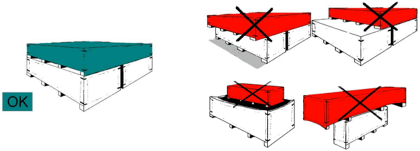
Semantics of programming languages

~~interpretation~~ ⟶ Logical formula

↓ execution = proof search

Observable property

Warehouse bin-packing



Box placements in containers:

- variables = box positions
- constraints = weight distribution, gravity. . .

Industrial partnerships with PSA, Fiat...

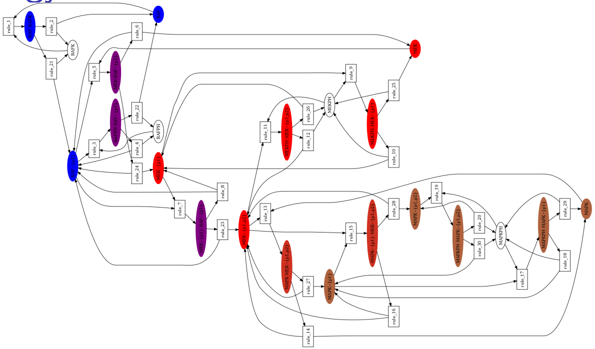
Optimizing energy in underground trains timetable

Reduce energy consumption by slight timetable shifting:

- variables = time shift
- constraints = energy limit

Industrial partnership with General Electrics

Analysis of large graphs of reaction networks in Systems Biology



Model analysis for conservation laws, dead-locks, comparisons between models.

- variables = molecules / vertices
- constraints = graph structure

Industrial partnership with Dassault Systme

Thesis

The design and the implementation of LCC

Design Selection of the right logical fragment of linear logic for modular programming, re-use of code, imperative traits...

Implementation The LCC compiler (and a compiler for a modular extension of Prolog), with efficient algorithms for proof search

Output Compiler for a new programming language for concurrent, imperative and constraint programming. Mono-paradigm: semantics driven by proof theory.

That's all folks!

Thank you!
Let's go for questions.