POEM

Partial Order Environment of Marseille

Peter Niebert
HongYang Qu

ELSE

- Réalisé avec Sarah
- Syntax à peu près sous-ensemble de IF 2.0
- Génération de code C « style CADP »
  - Avec « Event Zones »
  - Analyse préliminaire de dépendence
  - Génération inspirée par IF 1.0

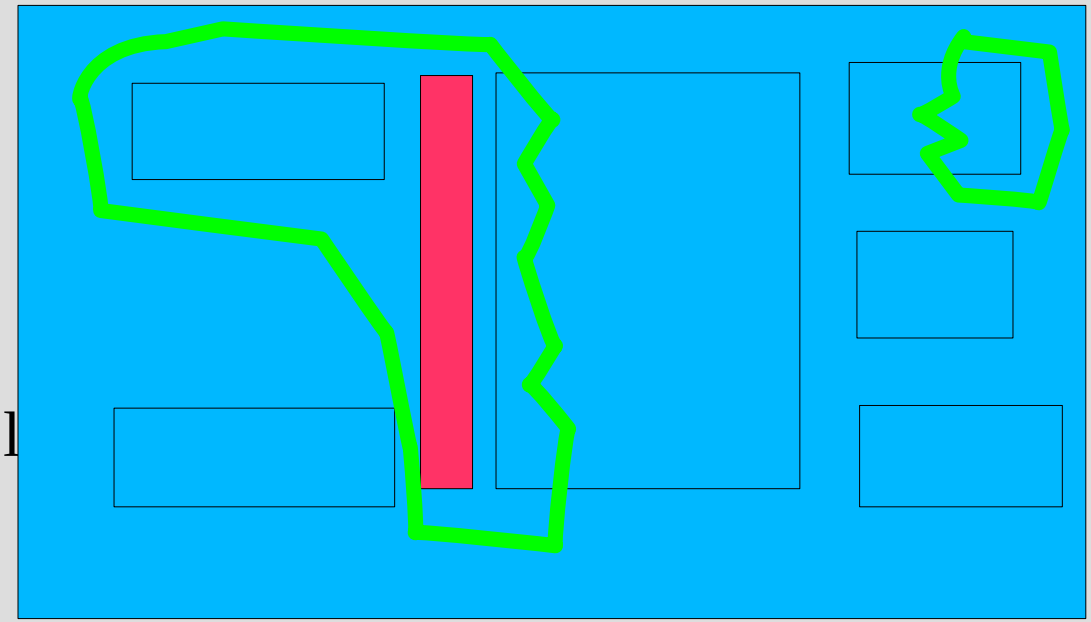
Why POEM

- ELSE was very incomplete
- Common problematic for Event Zones and LFS : « Mazurkiewicz Traces »
  - Static analysis of dependency was « ad hoc »
- Formula generation for « bounded model checking »?

Corner stones of « ELSE 2 » (situation one year ago)

Specification

Analysis method



IF 2

UppAal

Event zone automa

LFS etc

SAT solving

Corner stones of POEM (today)

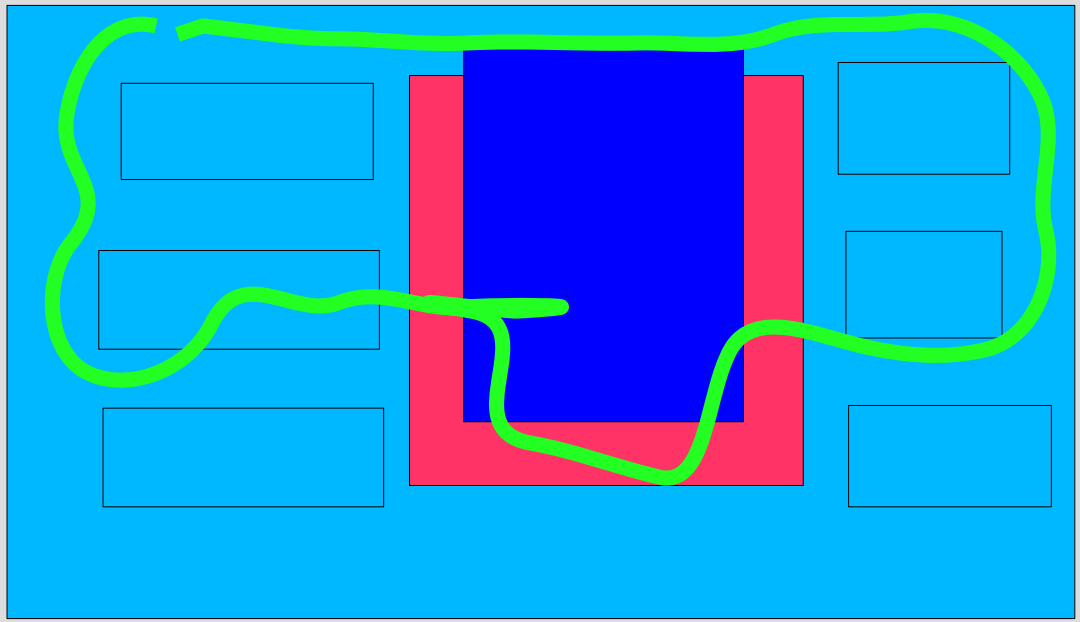
Specification

Analysis method

IF 2

Promela

UppAal



Event zones

LFS etc

SAT solving

Cornerstones of POEM

Specifications

IF 2

Promela

UppAal

Méthodes d'analyse

Event zone automa

LFS etc

SAT solving

« Eierlegende Wollmilchsau » ?

« couteau suisse? »

Cornerstones of POEM

- IF, Promela, UppAal:
  - Processes = automata with data
  - Instances of process templates
  - Complex data types
  - Clocks, timing constraints
  - Complex transitions (« while » programs)

Cornerstones of POEM

- Analysis with event zone automata
  - States = couples « discrete state », « zone »
    - Representation in C
  - Representation of implicit transitions as function types
    - For enabledness checking
    - For effect
  - Dependency analysis checking for « time progress » in event zones

Cornerstones of POEM

- LFS
  - Exploration with Mazurkiewicz traces

Cornerstones of POEM

- Bounded Model Checking
  - Code execution sequences as logical formulae
    « initial state, transition, state, ..., transition, desired state »
  - Structure of variables of such a formula
    - Variable nstances to represent each state in the sequence.
    - « Window » formulae for transitions
      - Maybe with auxiliary variables
  - Compression : **shorter sequences with multi steps**

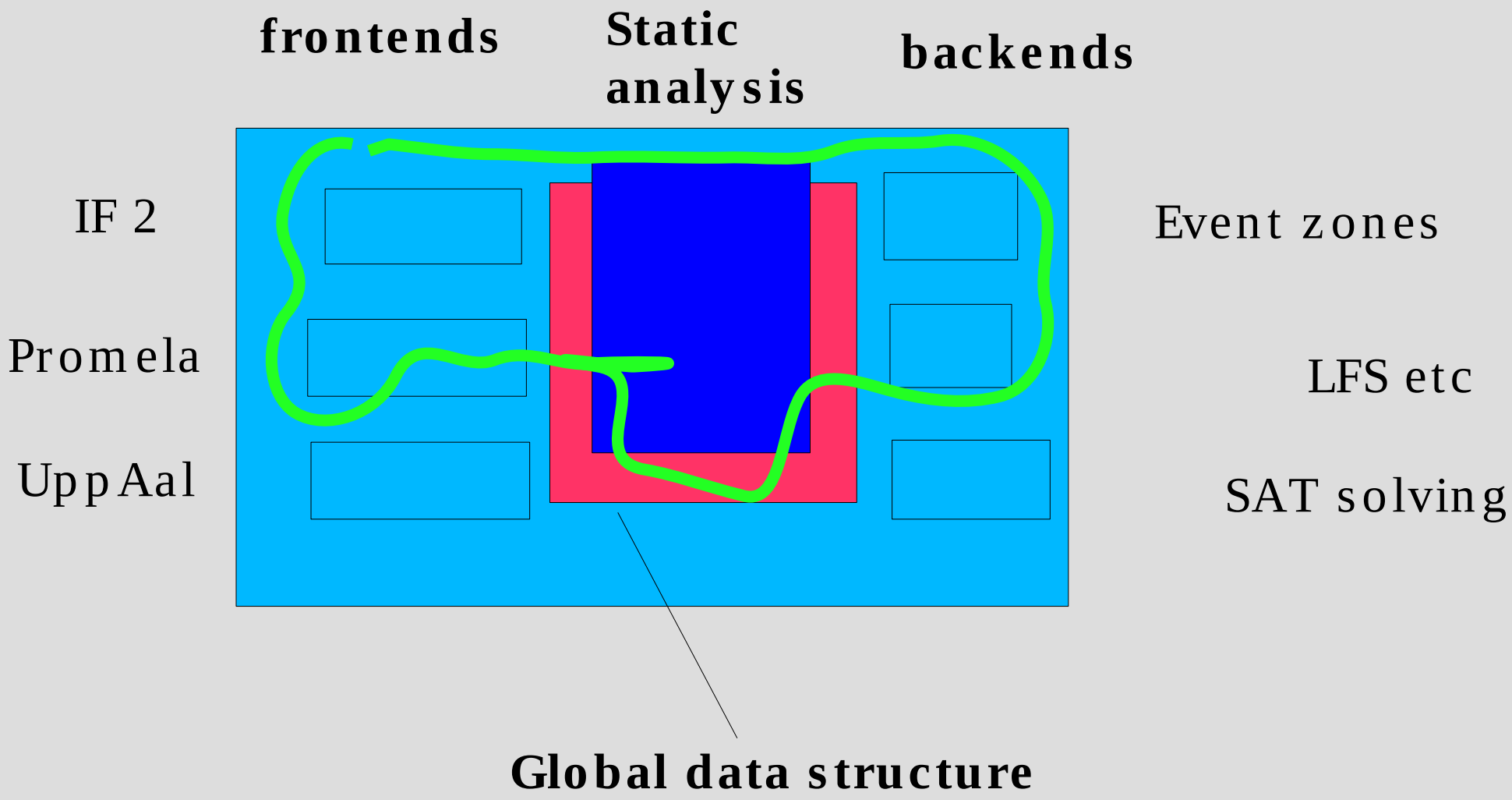
So why POEM

- Platform for experiments and **comparison** of algorithms
  - Eventually, we will be able to compare Spin/ UppAal directly with algorithms in POEM based on the same source specifications
- **« trace oriented »**, there aren't so many around ...

Talks

- Introduction (Peter)
- Particular topics
  - Architecture of POEM (Peter)
  - Independence in POEM (Peter)
  - An implementation of traces in POEM (HongYang)
  - Some updates on algorithms (HongYang)
  - Some problems for YOU

Architecture of POEM (OCaml)



Source structure

- Common
  - **Global data structure**, Rewriting, error handling ...
- Frontends
  - if2poem, promela2poem, ...
- Backends
  - Poem2c (different variants), poem2sat?
- Middle
  - Static analysis: type checking, ... **dependency**
- Chains
  - If2c, promela2sat, ...

Global Data Structure

- One data structure for
  - Specification
  - Symbol table, annotations
  - As interface for frontend and, partially backend
- Common services
  - Printing, rewriting, collecting information ...

Global Data Structure

- GDS for specification
  - (Indexed) Processes
    - Variables, types, constants ...
    - A behaviour given as complex automaton
  - Automata : states and transitions
  - Transitions :
    - Complex conditions, branching
    - Complex effect
- Effect :
  - « scoped while programs »

Independence in POEM

Basic model (currently)

- Transitions depend on variables and write to variables
- This gives a reader/ writer dependency
- The state of a process is a variable.
- Some variables « go together » with respect to dependency
- Static analysis: Identify « essential variables »

Representation of transitions

- Rewriting into a normal form :
  - Each transition is transformed to be deterministic, non-branching and gets a unique id
- To obtain this, we may have to split transitions. As a result, more transitions, but also more independence:
  - Each transition refers to a lower number of variables

Representations of dependency

- Dependency relation
- Clique cover of dependency relation
  - Number of cliques per transition? Smaller is better.
  - Minimal cover maybe NP-complete.
- Explicit read/write dependency
  - Number of (essential) variables per transition?

Challenges

- How to compute efficiently « small » distributed alphabets?
- Adequate order vs symmetry reduction
- We have a platform,
  - You have a problem?
    - Let's solve it!
  - You have an algorithm?
    - Let's integrate it!

GUI

- Work in progress
- Eclipse plugin
  - Editors for IF and Promela
  - GUI cover for tool chains
  - Not ready: Counter example animation
  - Planned as form of distribution for POEM
    - Platform dependent plugin, so POEM will be plug and play ;-)