A Tracer Driver for Versatile Dynamic Analyses of Constraint Logic Programs

Ludovic Langevine
SICS

In collaboration with Mireille Ducassé, IRISA

WLPE 2005
Sitges – October 2005

Outline

- Debugging needs in CLP(FD)
- Dynamic trace analysis
- Tracing driven by the analysis
- Assessment

CLP and Debugging

[Meier95,DiSCiPl97, …]

CLP is very declarative but

- Numerous variables and constraints (10^5)

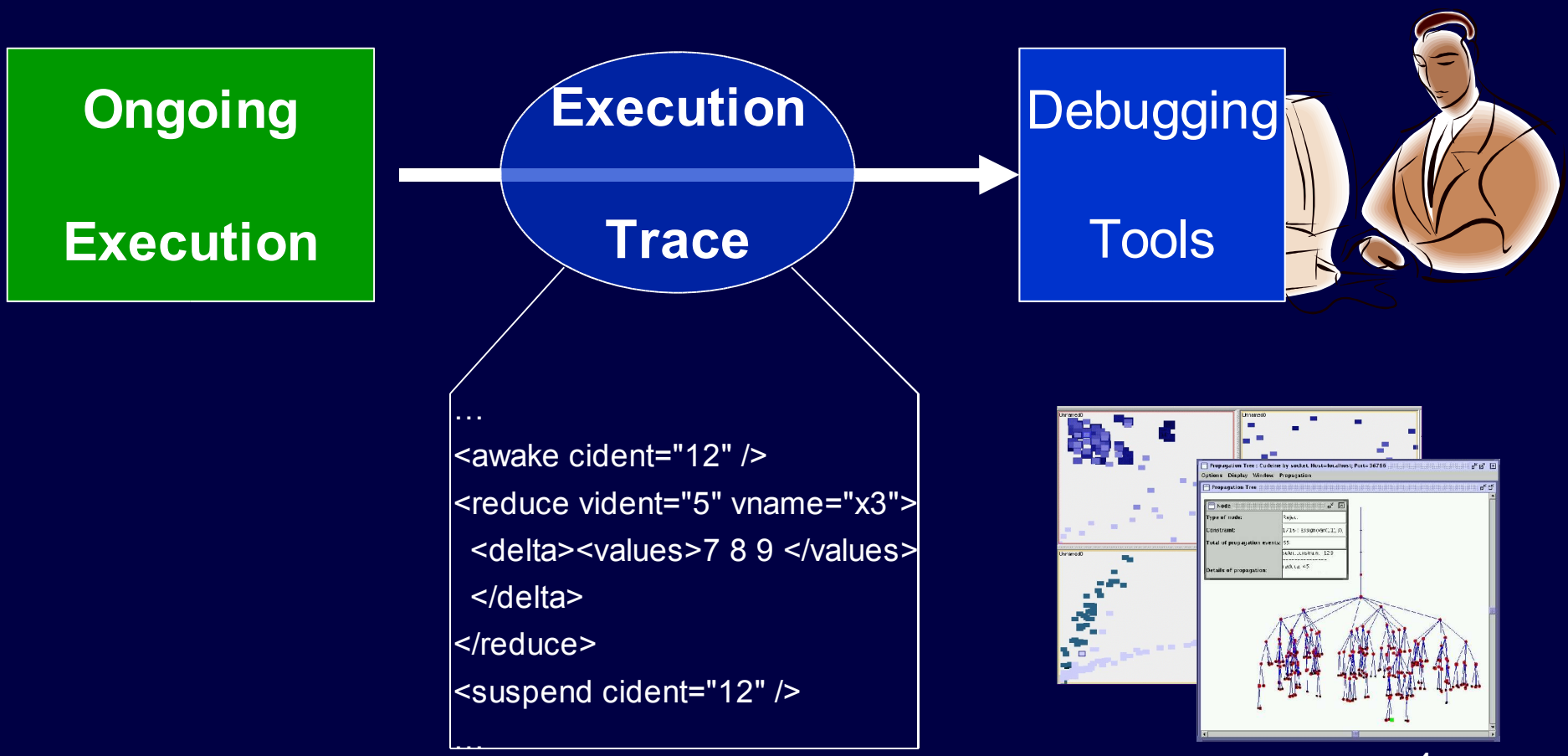
  $\Rightarrow$ What is the state of the system?

- Data flow embedded in the solver

  $\Rightarrow$ What is the behavior of the execution?

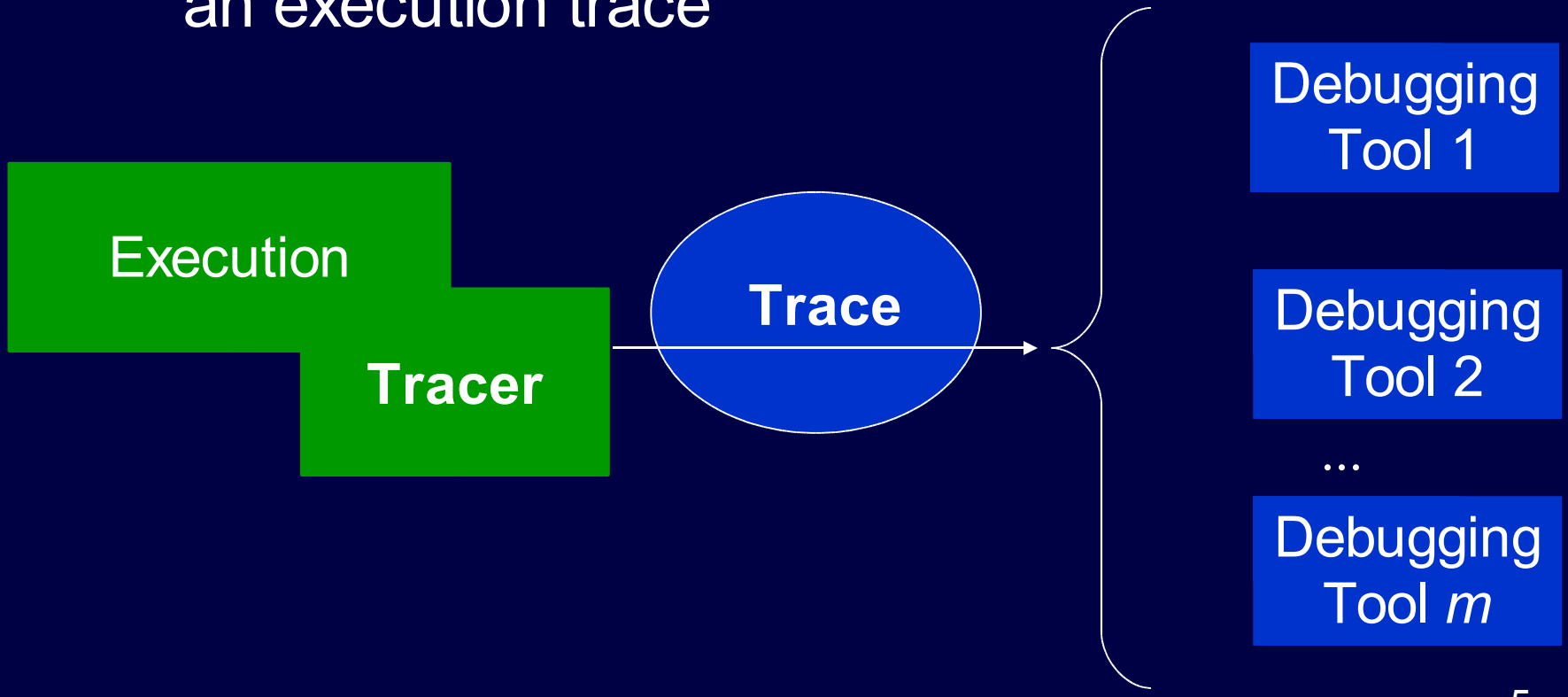
Specific debugging tools are needed

Our Proposal: an Execution Trace

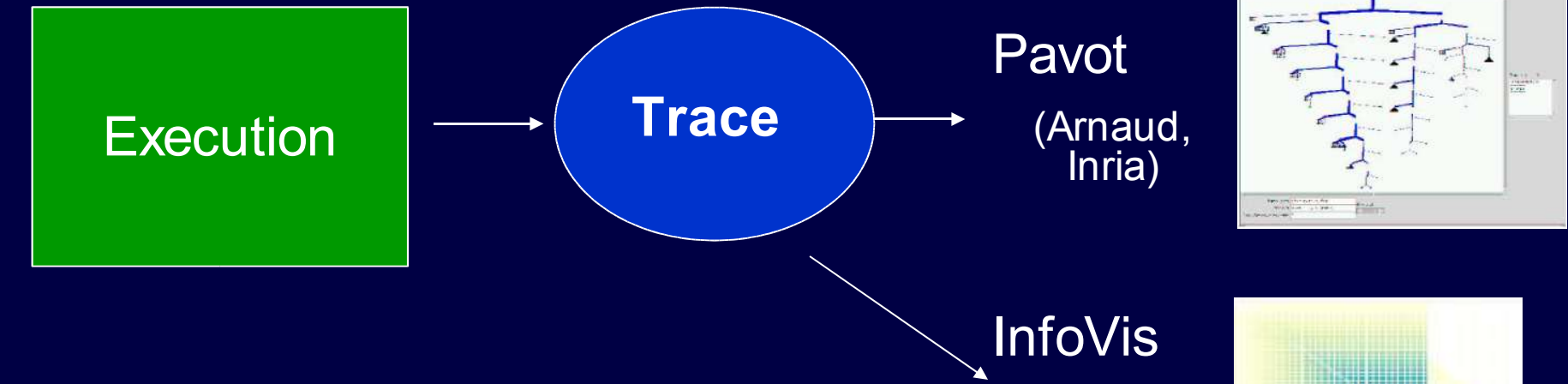


Guiding Principle

Versatile debugging tools can be built on top of an execution trace



For instance…



Key Issues

What is the content of the trace?

Define relevant events and attached data

(*trace schema*)

Pb.: Tools have versatile needs

⇒Trace has to be rich (Gentra4cp: 2GB per second)

How to produce the trace efficiently?

Pb.: *Overhead* ⇒ Non usability of the tools

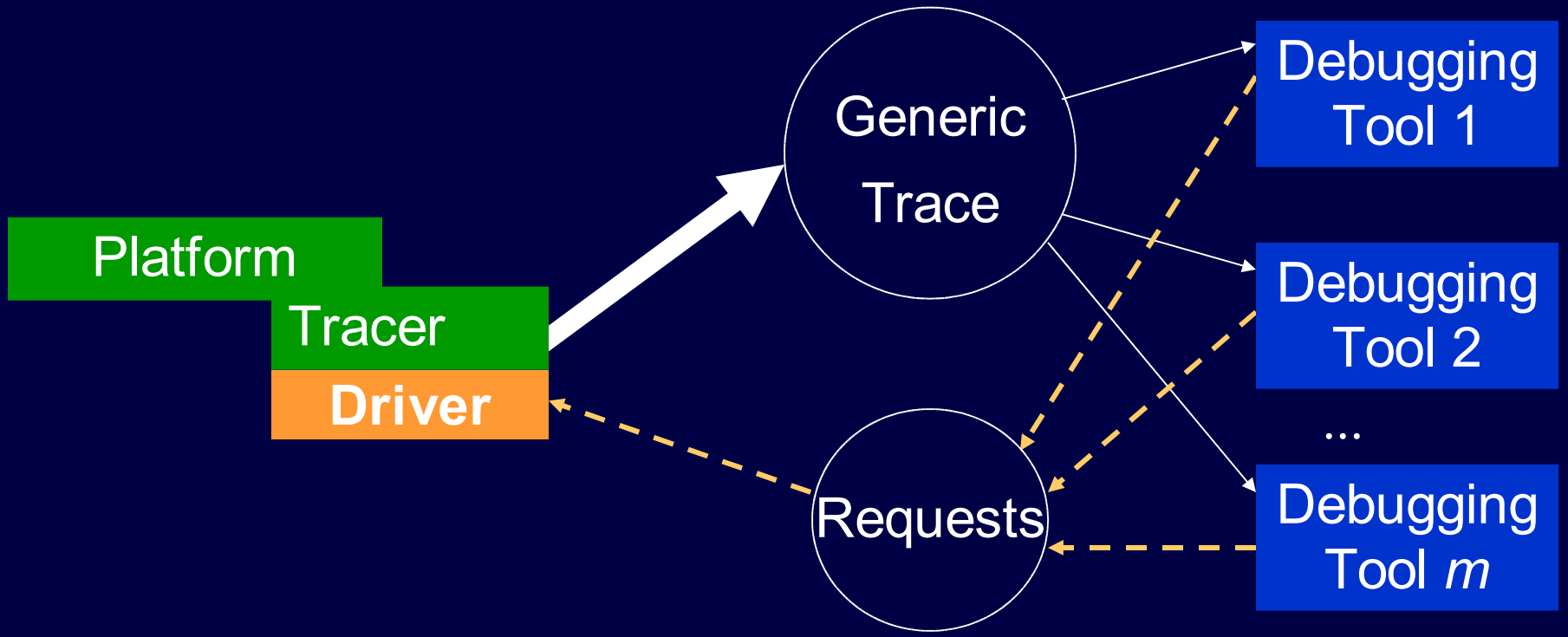
A compromise has to be found

A Rich Trace

- The trace is very rich (1s ≈ 2GBytes)
  - Costly to generate (tracer)
  - Costly to communicate (IPC)
  - Costly to process (debugging tool)
- A given tool needs only a small subpart of this huge trace

⇒ Adapt the trace to the needs of the tool: we propose a *tracer driver*

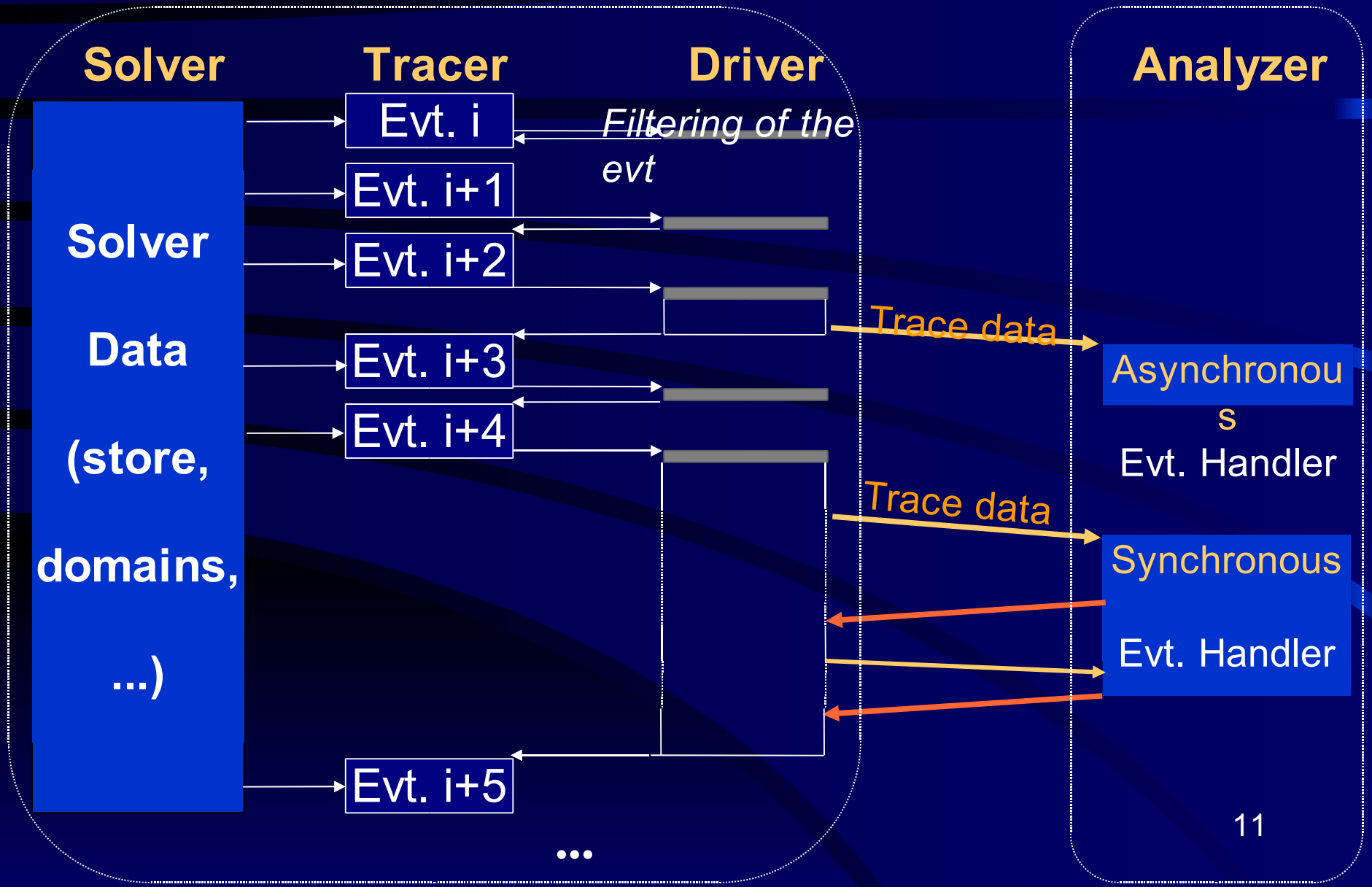
Rich *And* Efficient is Possible



Tracer driver

- A module of the tracer which drives the trace generation
- The tool describes its needs
  - ***event patterns***: ***When*** and ***What*** to trace
- The needs can be incrementally updated
  - Cope with the evolving needs of a tool
- Tracer and tool can be synchronized or not
  - Can investigate some execution states

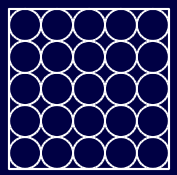
Principles of the Tracer Driver

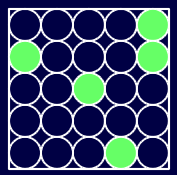


Event Patterns (1/2)

Full Trace —*predicate*→ Class of relevant events

Event Data —*selection function*→ Selection of trace data





Trace of the search-tree

search_tree: when port in [choicePoint, backTo, solution, failure] do current(**port, chrono, node**)

Event Patterns (2/2)

Contain a predicate on execution events

- Elementary condition on event attributes
  - Ex: constraint involved, variable, type of event, depth in the search-tree
  - Dedicated operators
- First order logic: negation, conjunction, disjunction
- Compiled into an automaton to evaluate the predicate

Primitive Commands

The driver can handle commands:

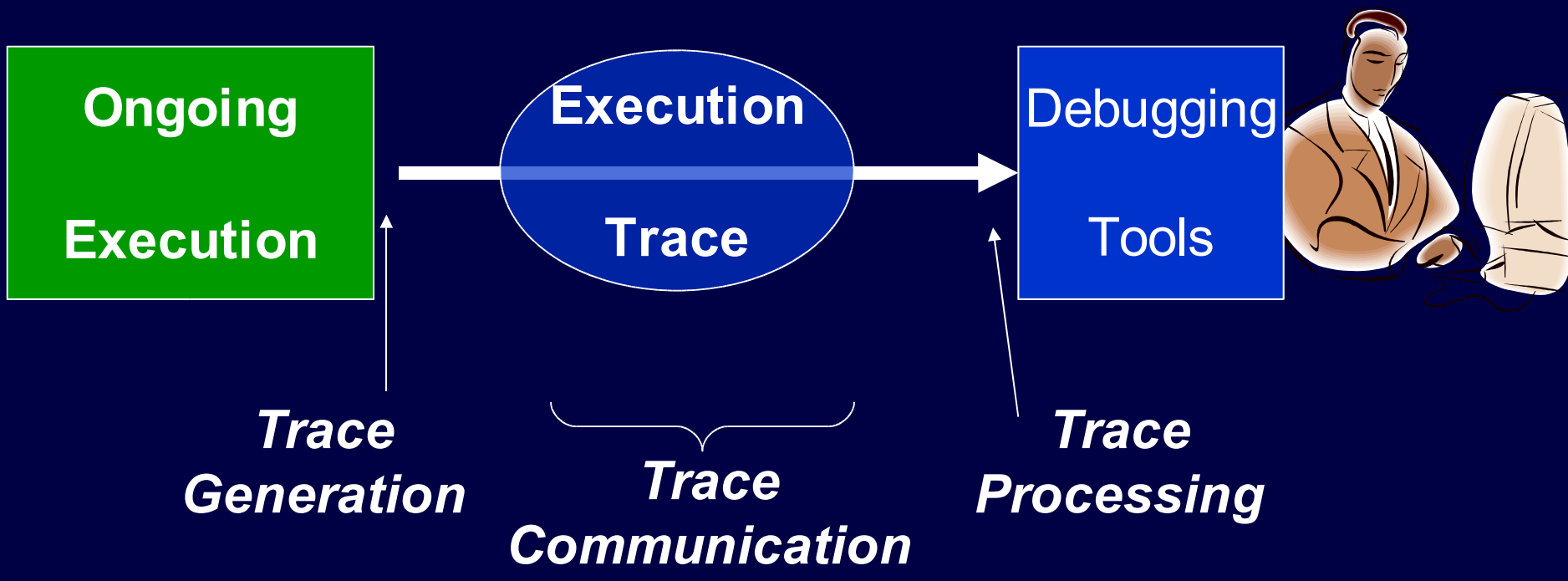
- RESET: reset all patterns
- ADD: add a pattern
- REMOVE: remove (disable) a pattern
- GO: resume the execution

$\Rightarrow$ Can be adapted to the evolving needs of the tools

Qualitative Assessment

- Is the tracer driver powerful?
  - Several existing architectures can be implemented in this framework (e.g. Opium [Ducassé92], Morphine [Jahier99])
  - Monitoring, debugging and visualization are enabled *in parallel*

Impact on Performances



Driver Performance

2 orders of magnitude better than the “generate and dump” architecture

- We pay only for what we need to trace
- The size of the trace is drastically decreased
  - Search-tree: 1/100

Its efficiency is inversely proportional to the mean duration of a trace event

OK for CP (a trace event ≈50ns)

The Tracer Driver

Is indeed a good compromise

- Rich trace possible
- Only the requested trace is generated
  - Reduces trace generation
  - Speeds up trace communication
  - Speeds up trace processing

Conclusion

- Development of dynamic tools is made easier
- Versatile analyses can be activated in parallel
- Synchronous ***and*** Asynchronous modes enabled
- No efficiency concern when defining the trace content
- The trace is generated on demand

A Tracer Driver for Versatile Dynamic Analyses of Constraint Logic Programs

Ludovic Langevine
SICS

Joint work with Mireille Ducassé, IRISA

WLPE 2005
Sitges – October 2005

Execution Data

- Execution data (trace)
  - Sequence of events of interest
  - Reflects the behavior of the execution
- **Trace schema** = definition of
  - relevant events
  - attached information