Kind-AI: When abstract interpretation and SMT-based model-checking meet

Pierre-Loïc Garoche – Onera – U. of Iowa
joint work with T. Kashai and C. Tinelli

04/13/2012

ONERA

THE FRENCH AEROSPACE LAB

retour sur innovation

Context: safety properties for controller

Open/Closed system analysis

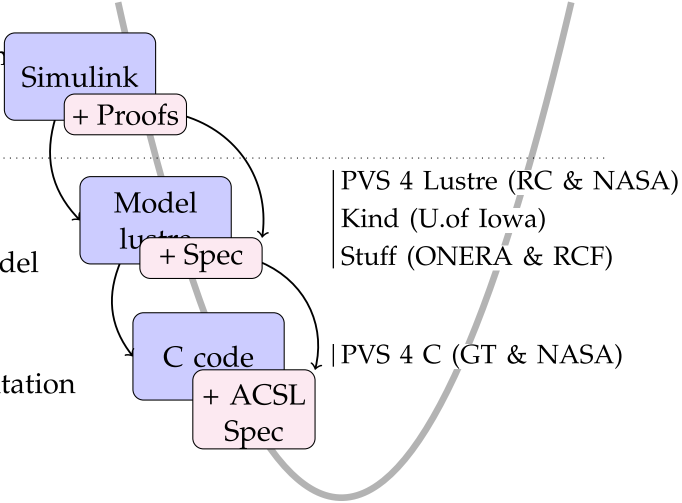
Control theorists

Computer scientists

Controller

Implementation model

Low level implementation



CONTEXT: SAFETY PROPERTIES FOR CONTROLLER

Open/Closed system analysis

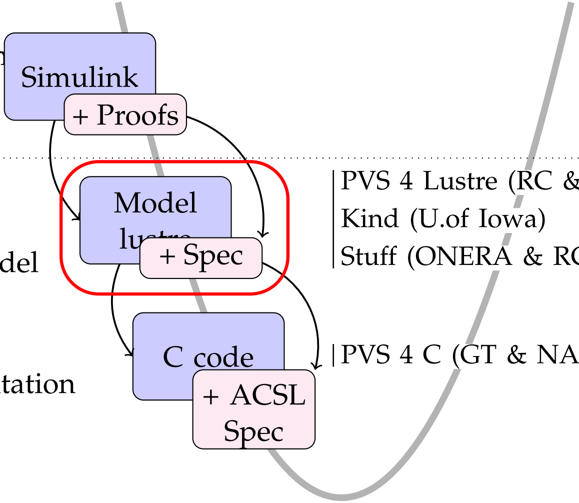
Control theorists

Computer scientists

Controller

Implementation model

Low level implementation



PVS 4 Lustre (RC & NASA)
Kind (U.of Iowa)
Stuff (ONERA & RCF)

PVS 4 C (GT & NASA)

Motivation

Motivation:

- prove a safety property over a transition system
- interested in numerical invariants

Available elements/Application

- k-induction engine for the transition system
- numerical abstract domains, ie. APRON
- application to Lustre models analysis

Numerical invariants

- Intervals
- Polyhedra
- Linear templates
- Linear expression under implication,

  eg. cond_1 and $\text{cond}_2 \implies$ linear expression

What for ?

- Identify an over-approximation of reachable states
  - prove target properties expressed as such invariants
  - enrich the description of the system by make explicit the implicit properties
  - or address more complex user-defined properties by considering only interesting states
- Constrains k-induction

Abstract interpretation

- Ideal approach to compute numerical invariants
- But . . .

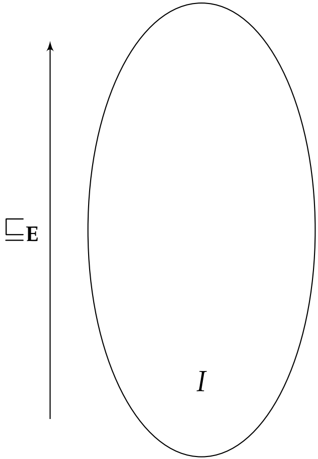
Abstract interpretation

- Ideal approach to compute numerical invariants
- But ...
  - results and time to get them depend on
    1. the abstraction used
    2. and speed-up parameters (widening, narrowing)

Abstract interpretation

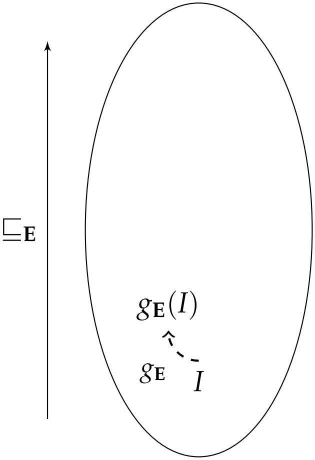
- Ideal approach to compute numerical invariants
- But …
  - results and time to get them depend on
    1. the abstraction used
    2. and speed-up parameters (widening, narrowing)
  - (could be) painful to define

Abstract interpretation – The usual picture



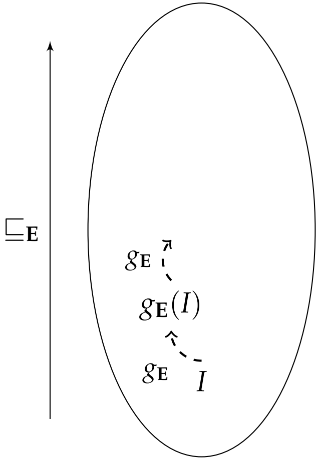
$\langle \mathbf{E}, \sqsubseteq_{\mathbf{E}} \rangle$
Set of formulas

Abstract interpretation – The usual picture



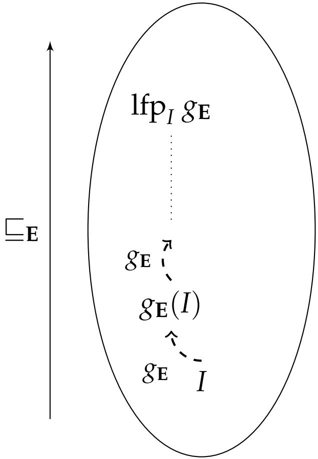
$\langle \mathbf{E}, \sqsubseteq_{\mathbf{E}} \rangle$
Set of formulas

Abstract interpretation – The usual picture



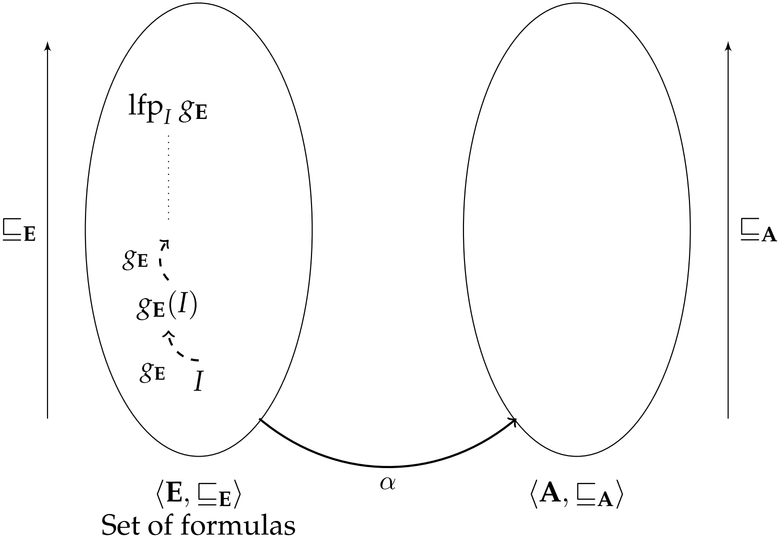
$\langle \mathbf{E}, \sqsubseteq_{\mathbf{E}} \rangle$
Set of formulas

Abstract interpretation – The usual picture



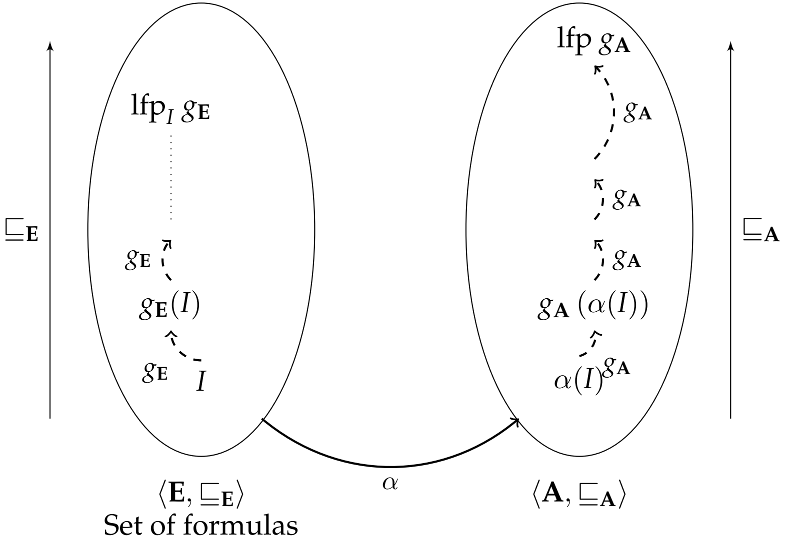
$\langle \mathbf{E}, \sqsubseteq_{\mathbf{E}} \rangle$
Set of formulas

ABSTRACT INTERPRETATION – THE USUAL PICTURE



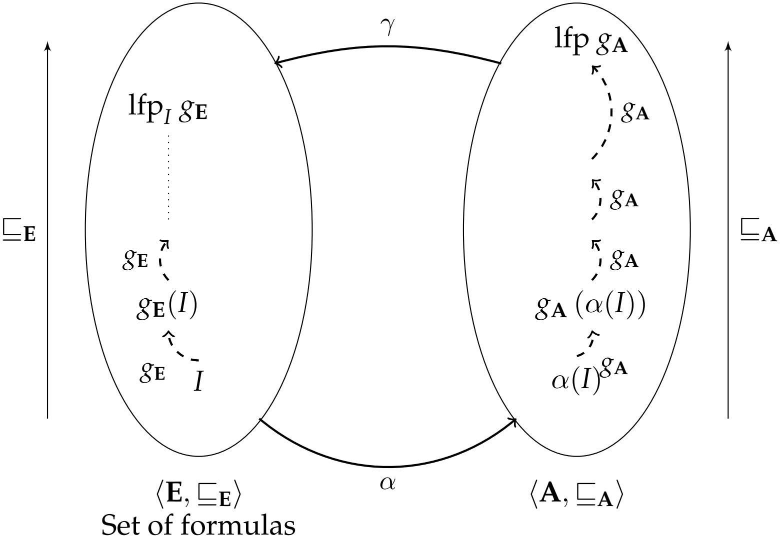
ONERA
THE FRENCH AEROSPACE LAB

ABSTRACT INTERPRETATION – THE USUAL PICTURE

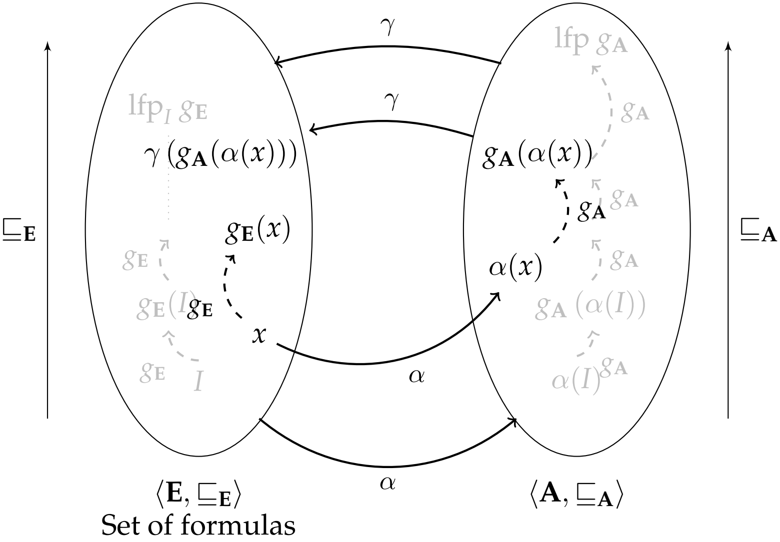


ONERA
THE FRENCH AEROSPACE LAB

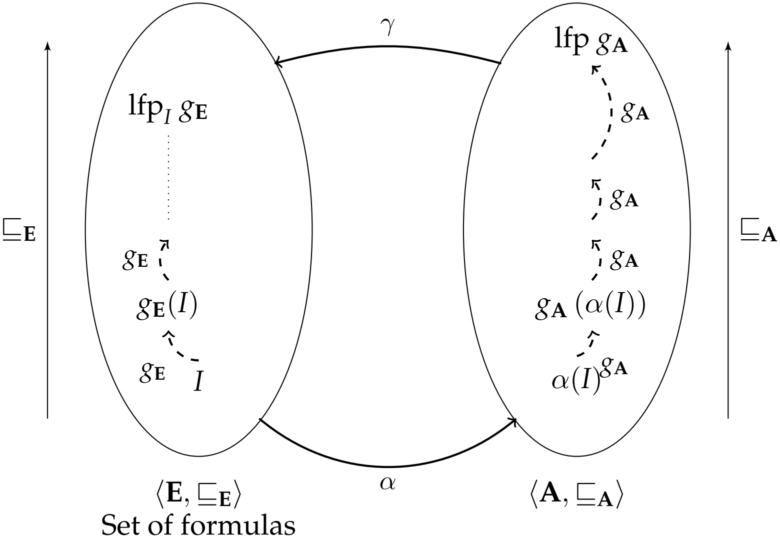
ABSTRACT INTERPRETATION – THE USUAL PICTURE



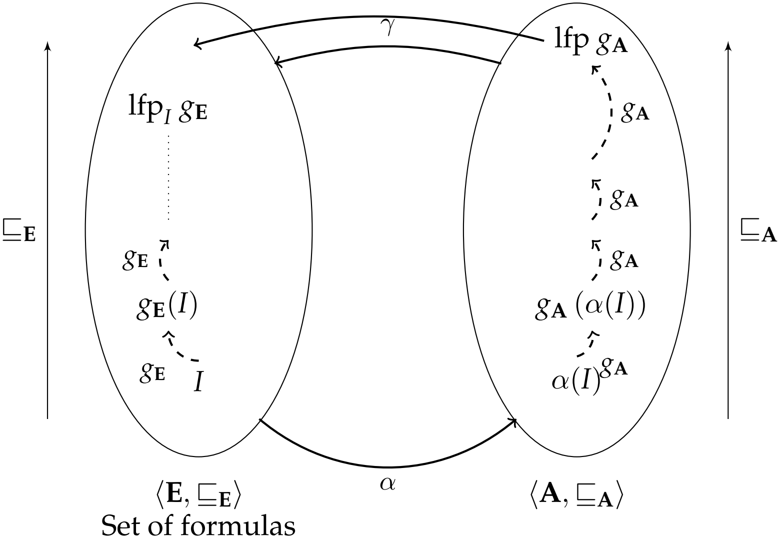
Abstract interpretation – The usual picture



ABSTRACT INTERPRETATION – THE USUAL PICTURE



Abstract interpretation – The usual picture



Basic ingredients

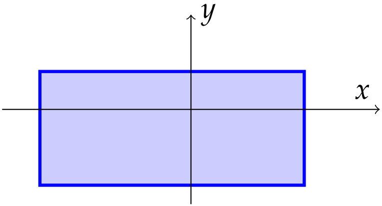
- Initial semantics expressed as fixpoint of a function $g_{\mathbf{E}} : \mathbf{E} \to \mathbf{E}$ over a lattice $\langle \mathbf{E}, \sqsubseteq_{\mathbf{E}} \rangle$. Easy for safety analysis: collecting semantics of a transition system $(\Sigma, I, \leadsto_T)$

$$\mathrm{lfp}_I \; \boxed{\lambda X. X \cup \{x' | x \in X, x \leadsto_T x'\}}$$

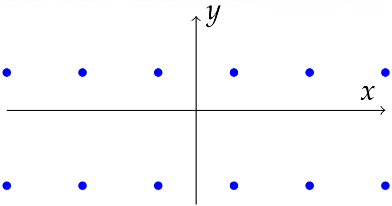
- Abstract representation of semantics values, here set of states: abstract domain $\langle \mathbf{A}, \sqsubseteq_{\mathbf{A}} \rangle$
- Relationship between original values and abstract ones, ie. a Galois connexion $\alpha : \mathbf{E} \to \mathbf{A}$ $\gamma : \mathbf{A} \to \mathbf{E}$
- Sound abstract transformers to mimic the concrete transitions in the abstract $g_{\mathbf{A}} : \mathbf{A} \to \mathbf{A}$

$$\mathrm{lfp}_{\alpha(I)} \, g_{\mathbf{A}}$$

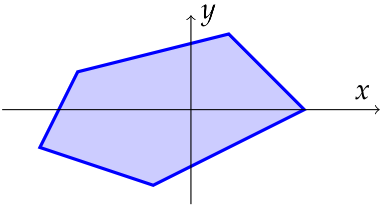
ABSTRACT DOMAINS



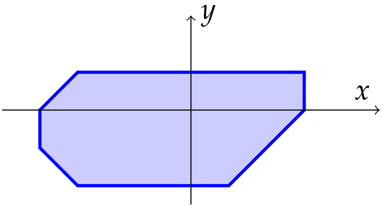
intervals



congruences



polyhedra



octagons

Abstract transformers

Usually the transition relation $\rightsquigarrow_T : \Sigma \to \Sigma$ is defined using smaller operators

- control flow ops: branching statements, loops, function calls, automaton transitions for FSM
- data flow ops: assigns of a variable, clock issues
- expression wise: depending on the available types, boolean operators, arithmetics operators, bitwise operators, or more complex data operators (arrays, trees, graphs, lists)
- memory wise: access to the value or the function of the pointer address
- etc …

Sound abstract transformers

- either the Galois connection **is implementable**. We can define a best transformer for each $op_{\mathbf{E}}$.

$$op_{\mathbf{A}_b}(a_1, ...a_n) = \alpha\left(op_{\mathbf{E}}\left(\gamma(a_1), \ldots, \gamma(a_n)\right)\right)$$

It is sound versus the Galois connection:
$\forall c_1, \ldots, c_n \in \mathbf{E}, a_1, \ldots a_n \in \mathbf{A}$

$$\forall i \in [1, n], c_i \sqsubseteq_{\mathbf{E}} \gamma(a_i) \implies op_{\mathbf{E}}(c_1, \ldots, c_n) \sqsubseteq_{\mathbf{E}} \gamma\left(op_{\mathbf{A}_b}(a_1, \ldots, a_n)\right)$$

Sound abstract transformers

- either the Galois connection **is implementable**. We can define a best transformer for each $op_{\mathbf{E}}$.

$$op_{\mathbf{A}_b}(a_1, ...a_n) = \alpha\left(op_{\mathbf{E}}\left(\gamma(a_1), \dots, \gamma(a_n)\right)\right)$$

  It is sound versus the Galois connection:
  $\forall c_1, \dots, c_n \in \mathbf{E}, a_1, \dots a_n \in \mathbf{A}$

$$\forall i \in [1, n], c_i \sqsubseteq_{\mathbf{E}} \gamma(a_i) \implies op_{\mathbf{E}}(c_1, \dots, c_n) \sqsubseteq_{\mathbf{E}} \gamma\left(op_{\mathbf{A}_b}(a_1, \dots, a_n)\right)$$

- or it **is not**: we have to produce some $op_{\mathbf{A}}$ satisfying the soundness condition.

  e.g. $op_{\mathbf{A}}(a_1, \dots, a_n) = \top_{\mathbf{A}}$ is sound.

What do we have ?

- A set of abstract domains provided by APRON
  - environment with intervals $x \mapsto [a, b]$, $y \mapsto [c, d]$
  - linear relations among variables (loose/strict polyhedra, octagons)
  - associated concretization function γ mapping abstract value to predicate of state variables in FOL: $\gamma(a)[x]$
- An axiomatisation of the system semantics $(\Sigma, I, \rightsquigarrow_T)$ expressed in FOL (targeting SMT)

$$I[x] \qquad T[x, y]$$

- An abstraction function from states to abstract elements: $\alpha_{\mathbf{Q}} : \Sigma \rightarrow \mathbf{A}$

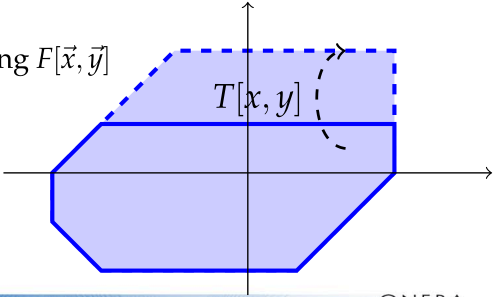
OBJECTIVE: AUTOMATIC ABSTRACT TRANSFORMERS

What do we want: generate automatically an abstract transformer for $op_{\mathbf{E}}$:
A sound function $op_{\mathbf{A}} : \mathbf{A} \rightarrow \mathbf{A}$ based on

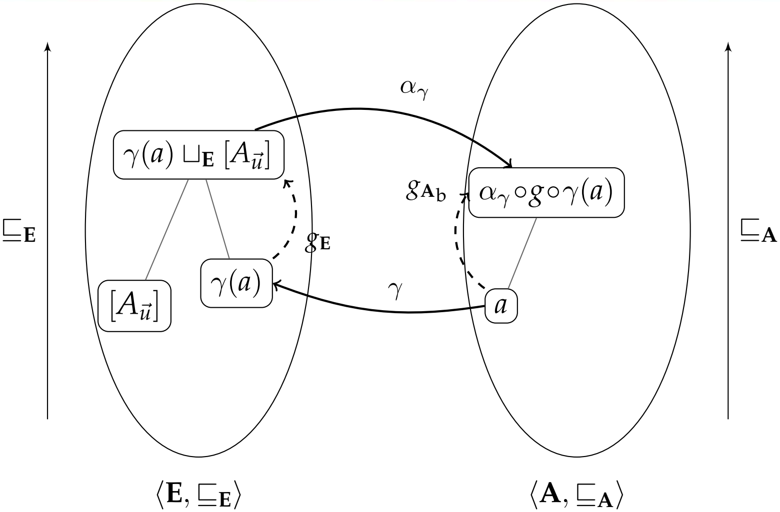
- the concretization function $\gamma : \mathbf{A} \rightarrow \mathbf{E}$
- the concrete operator $op_{\mathbf{E}} : \mathbf{E} \rightarrow \mathbf{E}$

The abstract transformer $g_{\mathbf{A}}$ maps an abstract state a to a bigger element describing more reachable states.

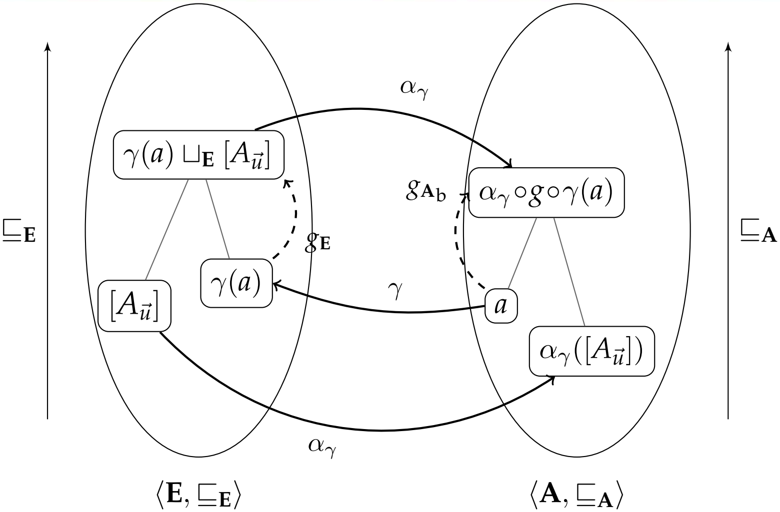
Input: $a \in \mathbf{A}$
$F[\vec{x}, \vec{y}] := \gamma(a)[\vec{x}] \wedge T[\vec{x}, \vec{y}] \wedge \neg\gamma(a)[\vec{y}]$
if F is unsatisfiable **then**
  return a
else
  let $\vec{v}, \vec{u}$ two states satisfying $F[\vec{x}, \vec{y}]$
  return $a \sqcup_{\mathbf{A}} \alpha_{\mathbf{Q}}(\vec{u})$



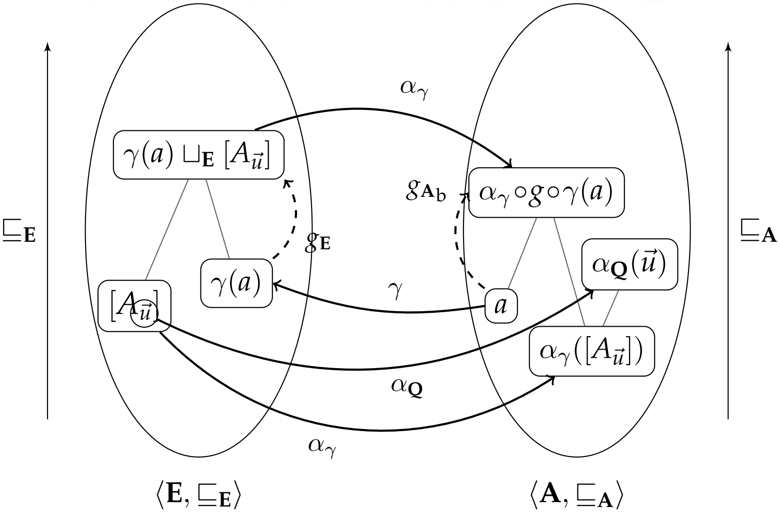
SOUNDNESS


α_γ
$\gamma(a) \sqcup_{\mathbf{E}} [A_{\vec{u}}]$
$g_{\mathbf{A}_b}$
$\alpha_\gamma \circ g \circ \gamma(a)$
$\sqsubseteq_{\mathbf{E}}$
$g_{\mathbf{E}}$
$\sqsubseteq_{\mathbf{A}}$
$\gamma(a)$
γ
a
$[A_{\vec{u}}]$
$\langle \mathbf{E}, \sqsubseteq_{\mathbf{E}} \rangle$
$\langle \mathbf{A}, \sqsubseteq_{\mathbf{A}} \rangle$

SOUNDNESS

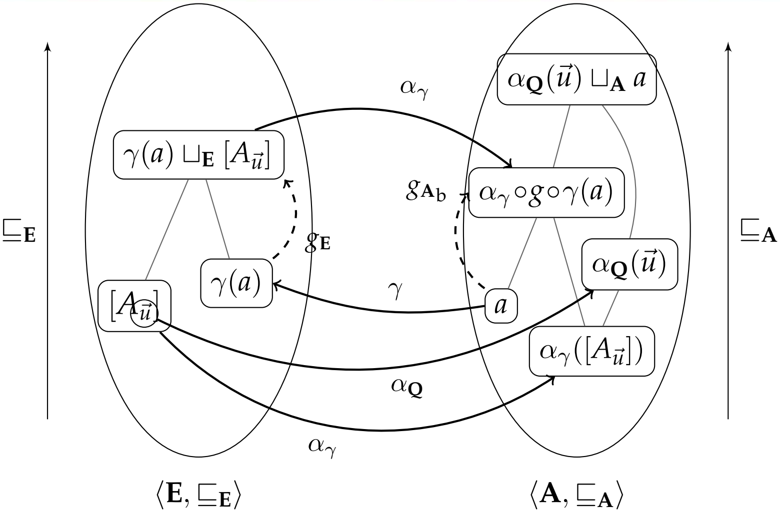

α_γ
$\gamma(a) \sqcup_{\mathbf{E}} [A_{\vec{u}}]$
$g_{\mathbf{A}_b}$
$\alpha_\gamma \circ g \circ \gamma(a)$
$\sqsubseteq_{\mathbf{E}}$
$g_{\mathbf{E}}$
$\sqsubseteq_{\mathbf{A}}$
$\gamma(a)$
γ
$[A_{\vec{u}}]$
a
$\alpha_\gamma([A_{\vec{u}}])$
α_γ
$\langle \mathbf{E}, \sqsubseteq_{\mathbf{E}} \rangle$
$\langle \mathbf{A}, \sqsubseteq_{\mathbf{A}} \rangle$

SOUNDNESS



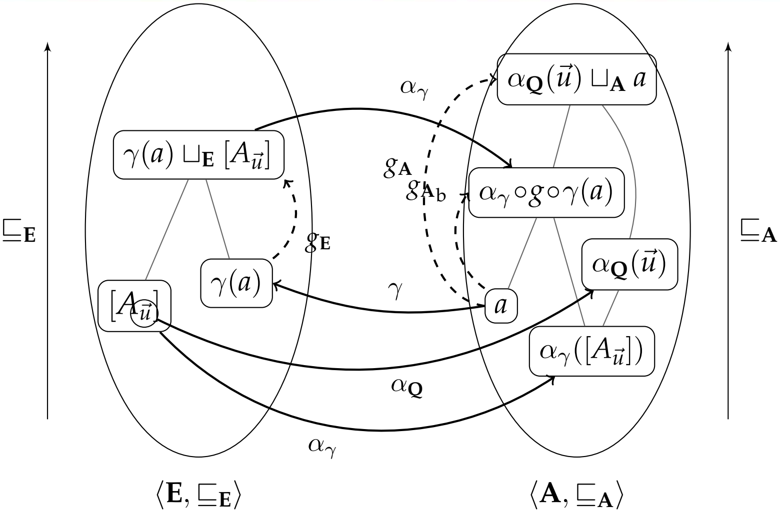
$\sqsubseteq_{\mathbf{E}}$
$\sqsubseteq_{\mathbf{A}}$
α_γ
$\gamma(a) \sqcup_{\mathbf{E}} [A_{\vec{u}}]$
$g_{\mathbf{A}_b}$
$\alpha_\gamma \circ g \circ \gamma(a)$
$g_{\mathbf{E}}$
$\alpha_{\mathbf{Q}}(\vec{u})$
γ
$\gamma(a)$
a
$[A_{\vec{u}}]$
$\alpha_\gamma([A_{\vec{u}}])$
$\alpha_{\mathbf{Q}}$
α_γ
$\langle \mathbf{E}, \sqsubseteq_{\mathbf{E}} \rangle$
$\langle \mathbf{A}, \sqsubseteq_{\mathbf{A}} \rangle$

Soundness



$\alpha_{\mathbf{Q}}(\vec{u}) \sqcup_{\mathbf{A}} a$
α_γ
$\gamma(a) \sqcup_{\mathbf{E}} [A_{\vec{u}}]$
$g_{\mathbf{A}_b}$
$\alpha_\gamma \circ g \circ \gamma(a)$
$\sqsubseteq_{\mathbf{E}}$
$g_{\mathbf{E}}$
$\sqsubseteq_{\mathbf{A}}$
$\alpha_{\mathbf{Q}}(\vec{u})$
$\gamma(a)$
γ
$[A_{\vec{u}}]$
a
$\alpha_\gamma([A_{\vec{u}}])$
$\alpha_{\mathbf{Q}}$
α_γ
$\langle \mathbf{E}, \sqsubseteq_{\mathbf{E}} \rangle$
$\langle \mathbf{A}, \sqsubseteq_{\mathbf{A}} \rangle$

SOUNDNESS



ONERA
THE FRENCH AEROSPACE LAB

The initial state abstraction

The fixpoint computation starts from an abstract of initial states.

$I_{\mathbf{A}} := \bot$
while $(I[\vec{x}] \wedge \neg\gamma(I_{\mathbf{A}})[\vec{x}]$ is satisfiable$)$ **do**
  let $\vec{v}$ be a state satisfying $I \wedge \neg\gamma(I_{\mathbf{A}})$
  $I_{\mathbf{A}} := I_{\mathbf{A}} \sqcup_{\mathbf{A}} \alpha_{\mathbf{Q}}(\vec{v})$
return $I_{\mathbf{A}}$

ONERA THE FRENCH AEROSPACE LAB

KIND-AI

The tool takes a Lustre model and generates numerical invariants

- uses all domains of APRON
- uses Kind front-end to parse Lustre and obtain the axiomatisation in SMT
- is parametric wrt the iteration strategies and widening threasholds
- is integrated with Kind to generate invariants but can be runned independantly
- open-source, written in OCaml

Kind-AI cont'd

Kind-AI can be parametrized by

- packing primitives: $(oct : x\ z)\ (poly : x\ y\ z)$
- partitioning primitives: $\{expr_1; expr_2 : packs\}$

Provided models will be injected in all partitions they satisfy

$$model \models_{\mathcal{L}} \neg expr_1 \wedge expr_2$$

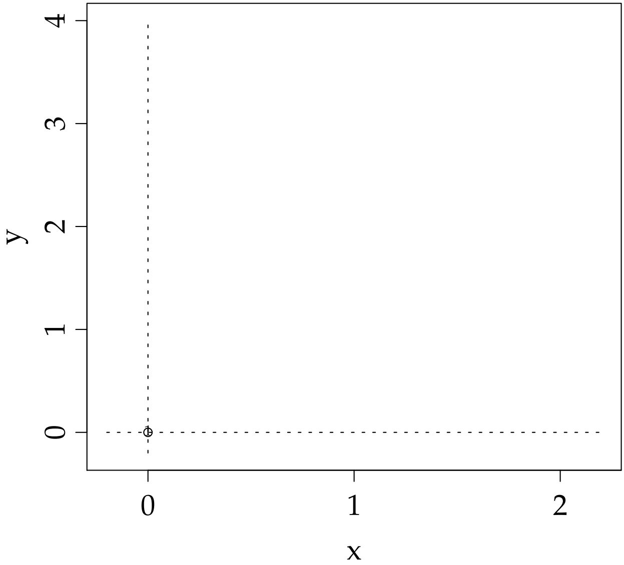
$\Longrightarrow$ model is injected in partitions $\neg expr_1 \wedge expr_2$.

Could also handle partitions over (small) finite range: $\{x : ()\}$ for x bounded.

Example

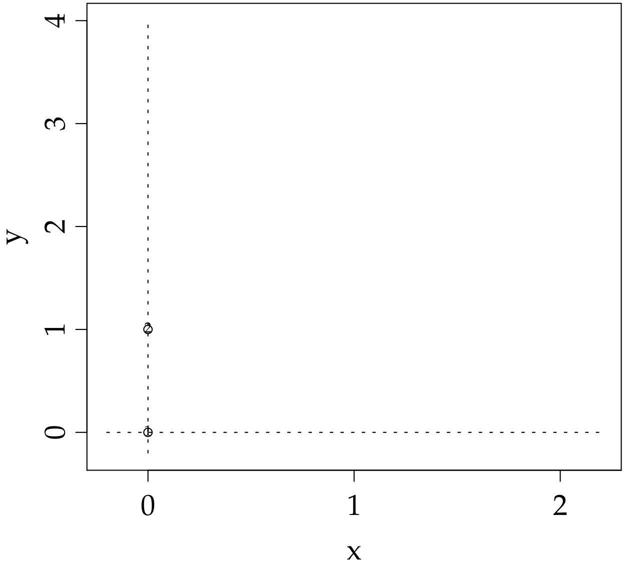
```
node parallel_counters (a,b,c:bool) returns(x,y: int; obs:bool);
var n1, n2:int;
let
  n1 = 10000; n2 = 5000;
  x = 0 -> if (b or c) then 0 else
           if a and (pre x) < n1 then (pre x) + 1 else pre x;
  y = 0 -> if c then 0 else
           if a and (pre y) < n2 then (pre y) + 1 else pre y;
  obs = (x = n1) implies (y = n2);
tel
```

Example cont'd

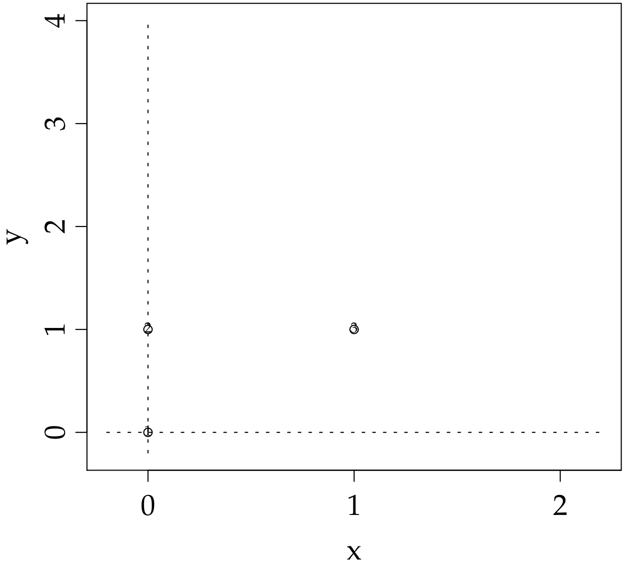


ONERA
THE FRENCH AEROSPACE LAB

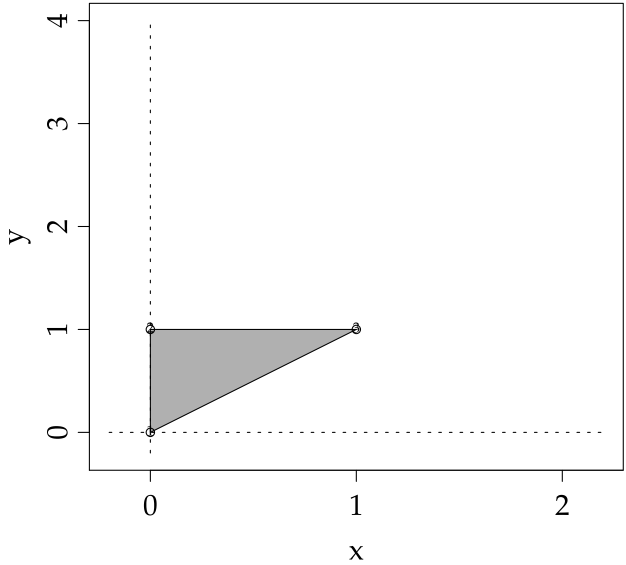
Example cont'd



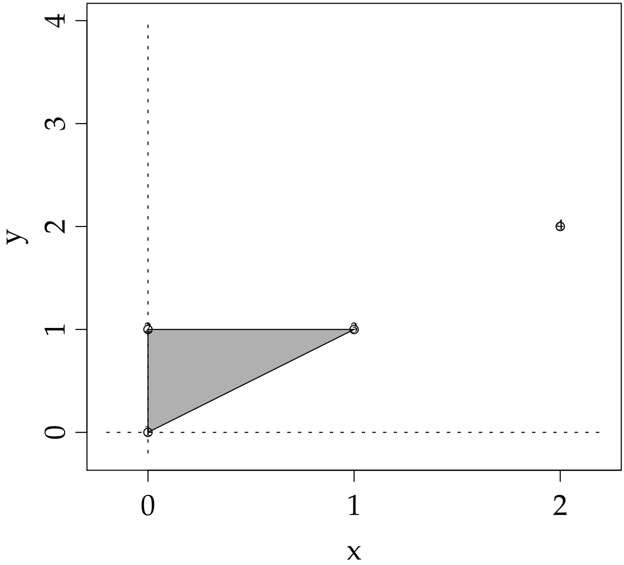
Example cont'd



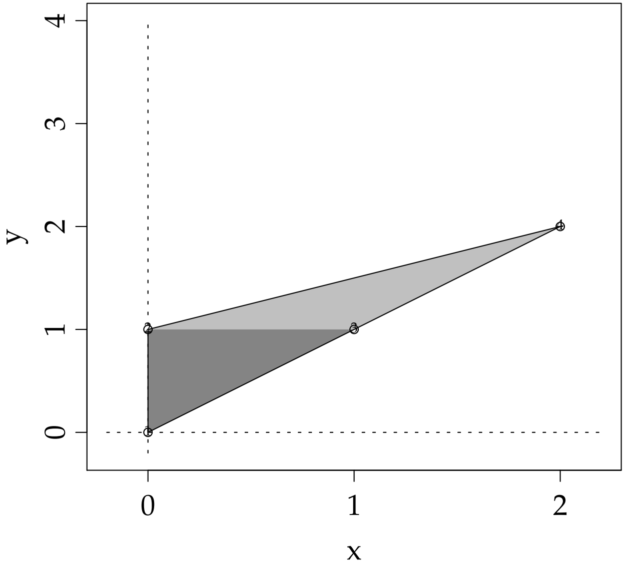
EXAMPLE CONT'D



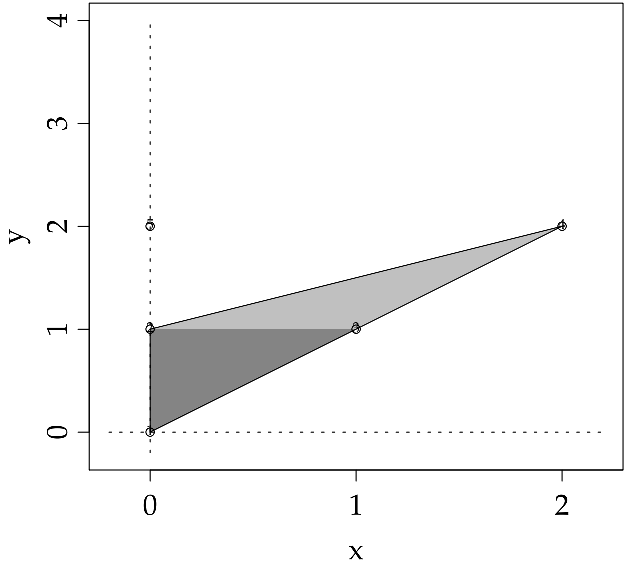
Example cont'd



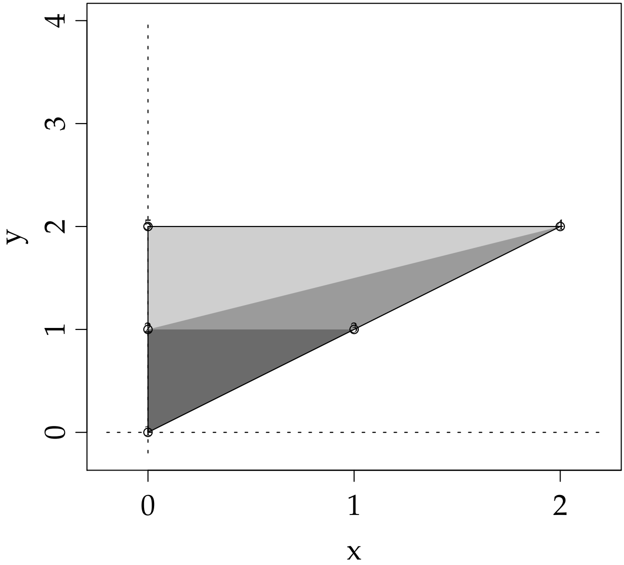
EXAMPLE CONT'D



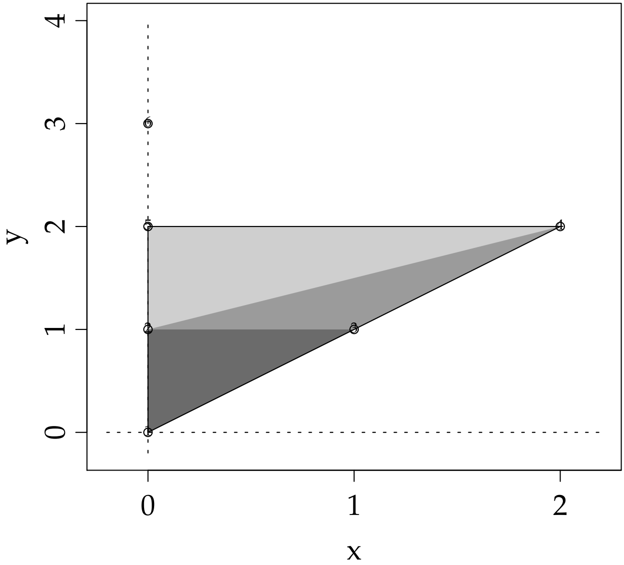
Example cont'd



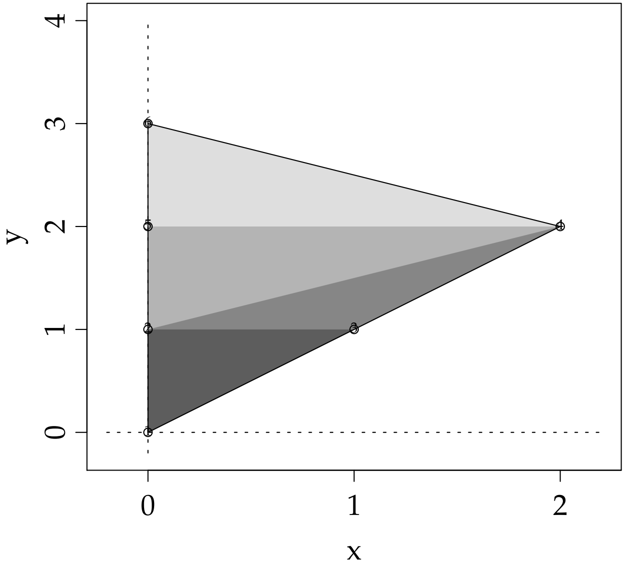
EXAMPLE CONT'D



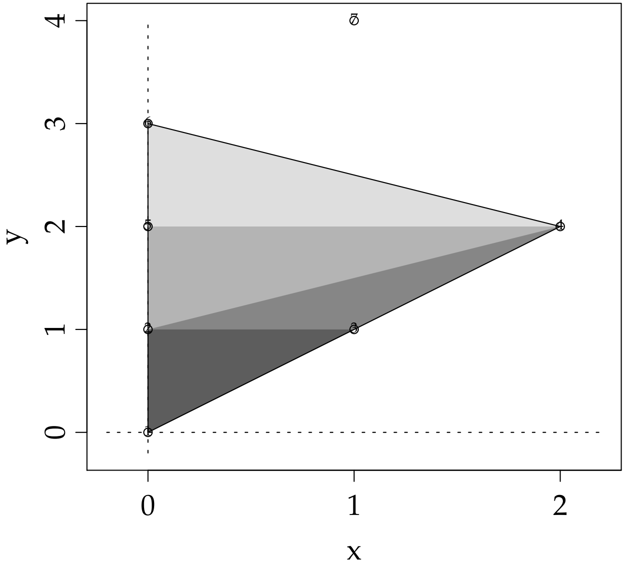
EXAMPLE CONT'D



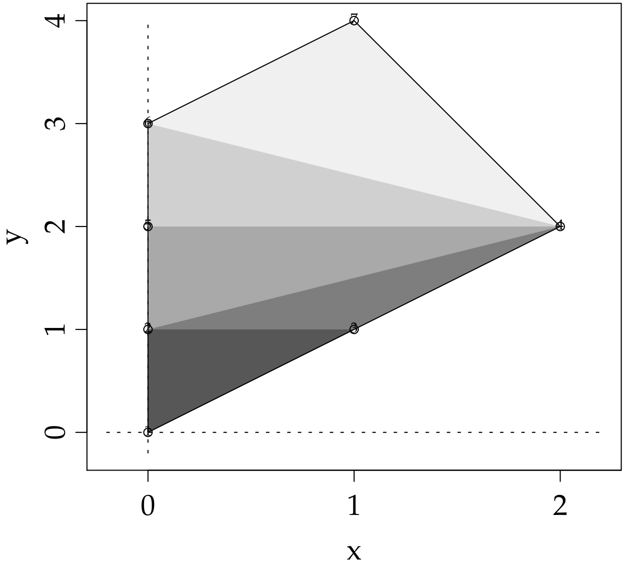
Example cont'd



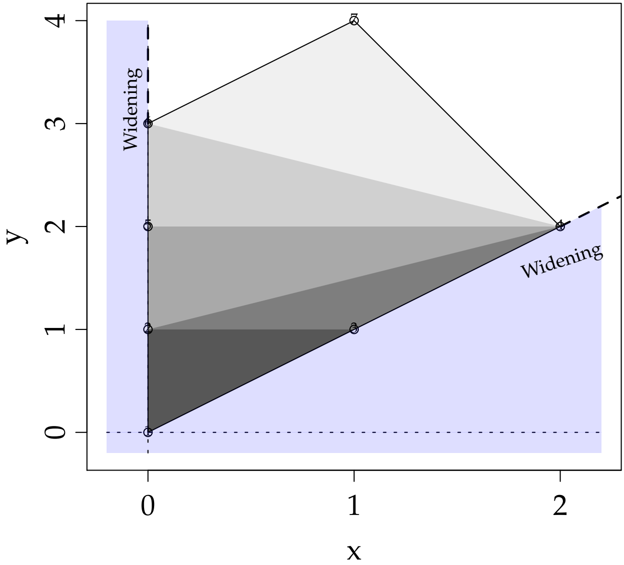
Example cont'd



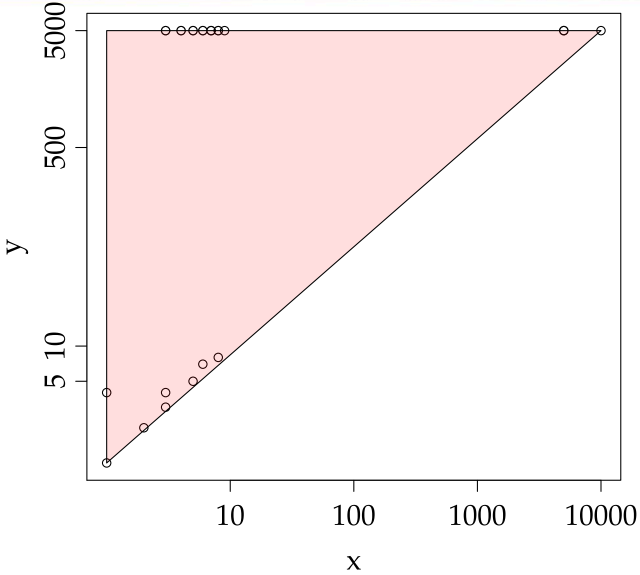
EXAMPLE CONT'D



Example cont'd



Example cont'd



Early invariants

Using a multiproperty technique for induction, concretization is expressed as a conjunction of identified sub-formula:

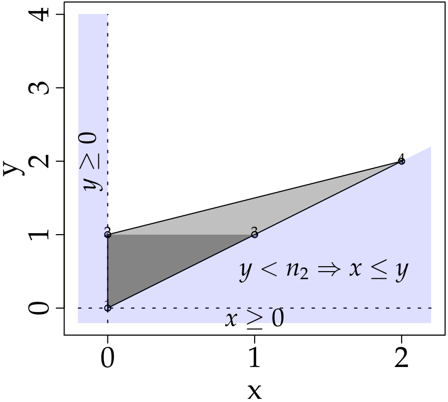
$$\gamma(a) = P_1 \wedge \ldots \wedge P_n$$

At each iteration of the fixpoint computation, we identify stable subparts, ie. invariants.

Example: $x \in [a, b]$ is concretized to $x \geq a \wedge x \leq b$. For increasing values of x, $x \geq a$ can be produced before the fixpoint is reached.

Kahsai, Garoche, Tinelli and Whalen. Incremental verification with mode variable invariants in state machines

Example cont'd



At the fourth iteration, the following properties are proven:

- $x \geq 0$ $\quad y \geq 0 \quad$ $y < n_2 \implies x \leq y$

Conclusion

- A generic approach for synthetizing abstract interpreters
  - needs the encoding of the transition systems in logic with entailment
  - and abstract domains that can be concretized to this logic
- Instanciation on Lustre models analysis
  - APRON domains
  - Kind k-induction Lustre axiomatization
- Generates a flow of (guarantied) invariants <u>before</u> reaching the final fixed point.