"About" 20 Years of AI in Italy

Francesco Ranzato
University of Padova

The Beginnings in Italy: Where

Pisa

Padova

The Beginnings in Italy: When and Who

1991: First published papers in international venues

Giorgio Levi
group in Pisa

Gilberto Filé
group in Padova

The Beginnings in Italy: What

- Static analysis of logic programming (LP) by abstract interpretation
- LP was popular in early 1990s and static analysis of LP was (and still is) a key concern
- Main Italian contributions to LP analysis:
  - Abstract declarative semantics of LP: bottom-up, top-down, compositional/modular, dealing with cut, Prolog, constraints, concurrency
  - Analysis of: variable sharing/aliasing, groundness, freeness, independence, numerical variables in CLP, etc.

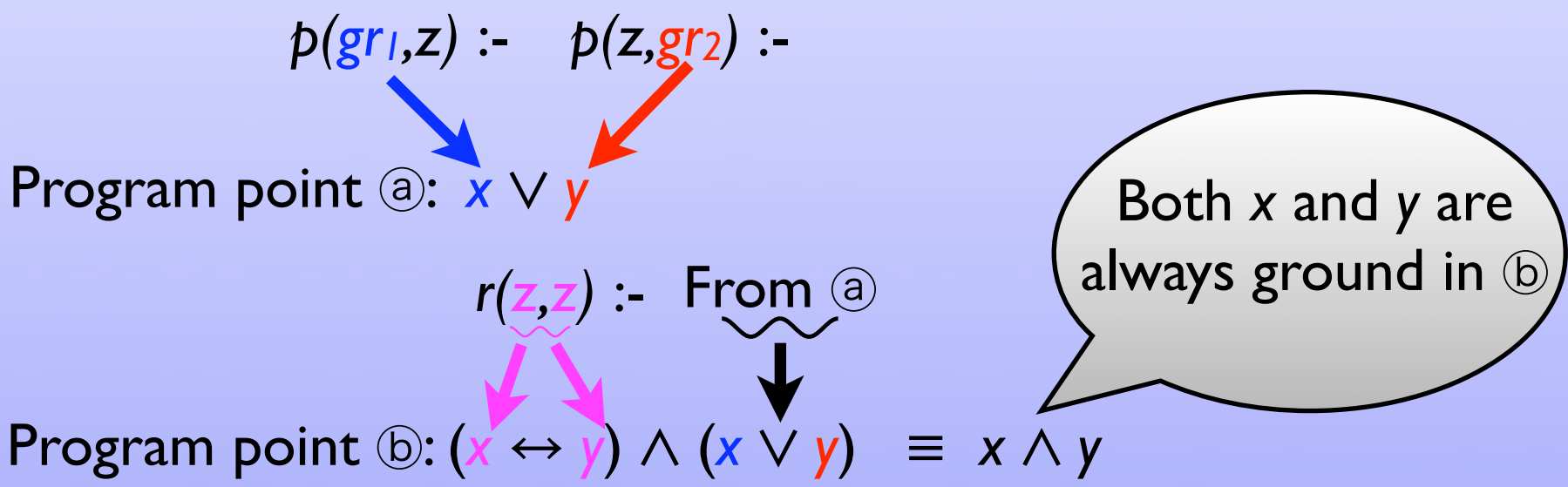
Groundness Analysis of LP

- At a given program point, is a logical variable *x* always bound to ground terms?
- Useful for speeding-up unification, avoiding occur-check, improving on/combining with other analyses (independence, sharing)
- Abstract domains of Boolean functions: $x \rightarrow y$ means "whenever *x* becomes ground so does *y*"

Groundness Analysis of LP

$q(x,y)$:- $p(x,y)$ ⓐ $r(x,y)$ ⓑ

$p(gr_1, z)$:-
$p(z, gr_2)$:-
$r(z,z)$:-

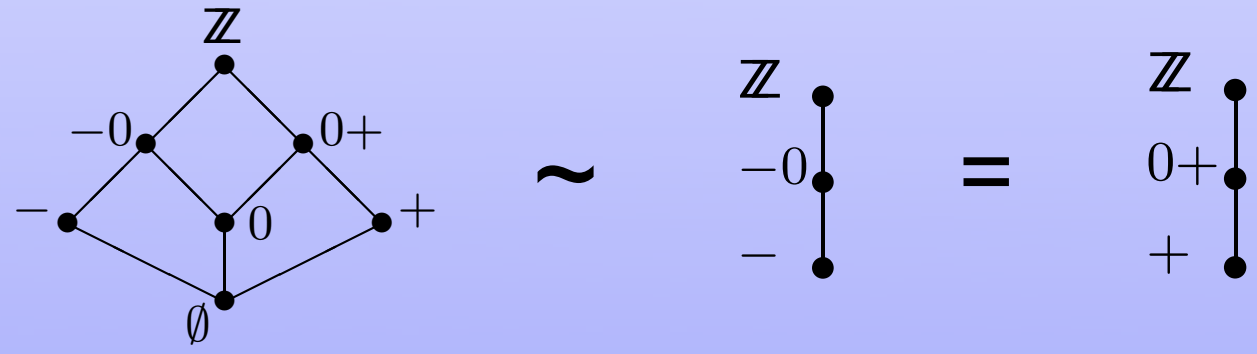


The 1990s in Italy

- New LP analyses: Abstract diagnosis, AI-based verification, type analysis, analysis of concurrent LP
- More precise and efficient abstract domains for LP: combined sharing/aliasing analysis, set vs pair-sharing, precise widening, improved abstract unification
- Systematic design/transformation of abstract domains: complementation, functional/linear refinement, disjunctive simplification, compression, complete refinement/simplification

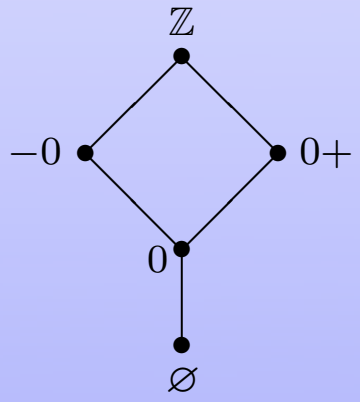
Abstract Domain Complementation

- Reduced product: $A_1 \sqcap A_2$ is the most abstract domain that is more precise (i.e. contains) both A_1 and A_2
- The inverse of reduced product: $A_1 \sim A_2$ is the most abstract domain whose reduced product with A_2 gives back A_1



Completeness in AI

- Concrete semantics $f : C^n \to C$
- Abstract semantics $f^\# : A^n \to A$ is complete when $\alpha \circ f = f^\# \circ \gamma$
- Concrete semantics: addition/multiplication on sets of integers
- Abstract domain of Signs



- Abstract addition $+^{Signs}$ is not complete

$\alpha(\{-1\} + \{2\}) = 0+$
$\alpha(\{-1\}) +^{Signs} \alpha(\{2\}) = -0 +^{Signs} 0+ = \mathbb{Z}$

- Abstract multiplication $\times^{Signs}$ is complete

Completeness Transformers

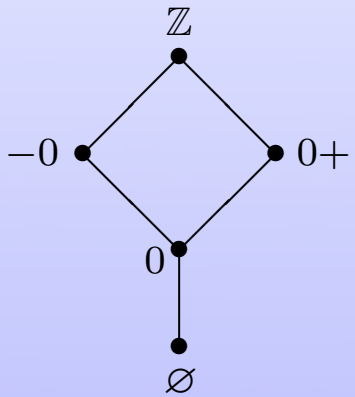
- Completeness is an abstract domain property: the possibility of defining a complete abstract semantics depends only on A

- Question: Can we minimally transform (enlarge/reduce) an abstract domain in order to make it complete? YES!

  ...CONSTRUCTIVELY!

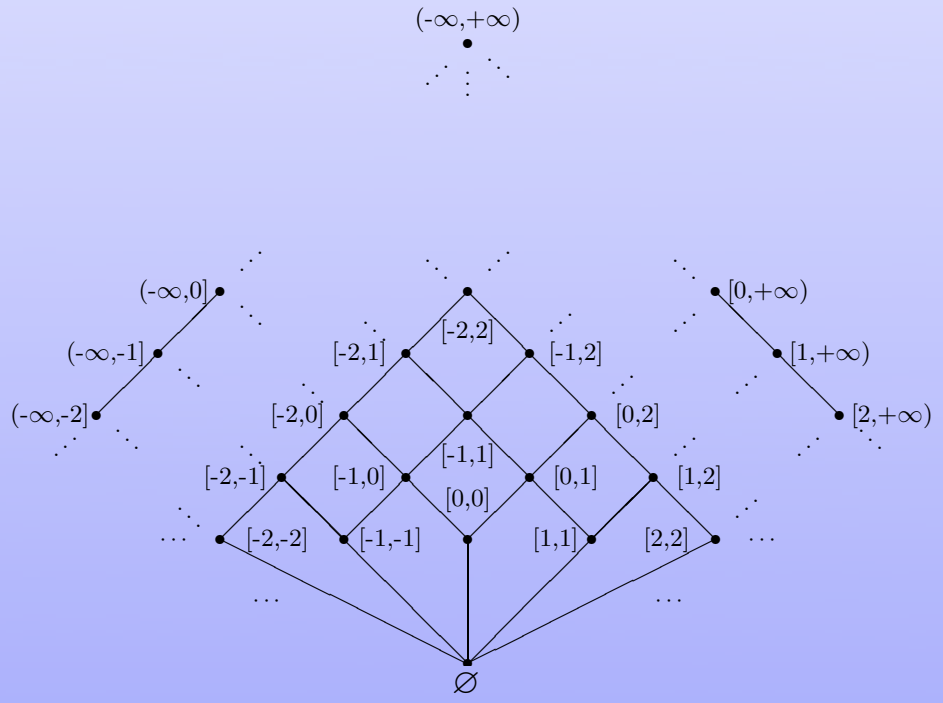
Completeness Transformers

- Signs is not complete for +



- Minimal complete restriction of Signs

- Minimal complete enlargment of Signs: Intervals



The 2000s in Italy

- A larger AI community: Padova, Parma, Pisa, Venice, Verona
- Numerical abstract domains, polyhedra, numerical widenings
- AI for non-interference and information flow in programming languages
- AI for abstract model checking and behavioural equivalences
- AI for code obfuscation and malware detection
- AI for analyzing OO programs
- AI for analyzing concurrent/distributed/mobile process calculi
- Probabilistic AI

Abstract Non-Interference

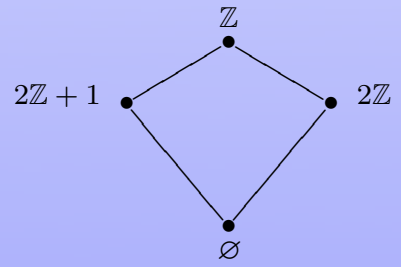
- Non-interference: $\forall \ell \in \text{Low}, h_1, h_2 \in \text{High}, \text{Low}(P(\ell, h_1)) = \text{Low}(P(\ell, h_2))$

```
while h>0 do {
  ℓ=ℓ+2; h=h-1;
}
```

  - ℓ changes depending on the initial value of h, i.e. implicit flow from h to ℓ: P is not secure
  - However, if the attacker only observes parity information, P is secure

- Abstract non-interference
  A,B abstract domains, P is (A,B)-secure if $\forall \ell_1, \ell_2 \in \text{Low}, h_1, h_2 \in \text{High}$,
  $\alpha_A(\ell_1) = \alpha_A(\ell_2) \Rightarrow \alpha_B(\text{Low}(P(\ell_1, h_1)) = \alpha_B(\text{Low}(P(\ell_2, h_2)))$

- Parity abstract domain



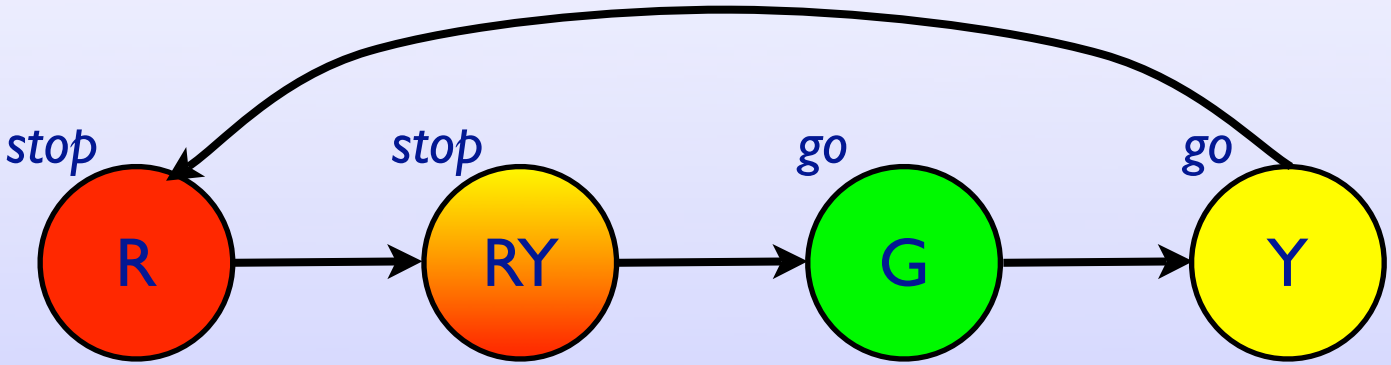
  - P is (Parity,Parity)-secure

Strong Preservation in Model Checking

- Kripke structures $\mathcal{K} = \langle \Sigma, \rightarrow, \ell \rangle$ as system models
- Abstract Kripke structures $\mathcal{A} = \langle A, \rightarrow^{\#}, \ell^{\#} \rangle$ as abstract models
  - A abstract state space, surjection $h: \Sigma \rightarrow A$, i.e. A is a partition of Σ
  - $\rightarrow^{\#}$ abstract transition relation on A
- Strong preservation for a language $\mathcal{L}$:

$$\forall \varphi \in \mathcal{L},\ \mathcal{K} \models \varphi \Leftrightarrow \mathcal{A} \models \varphi$$

Strong Preservation in Model Checking

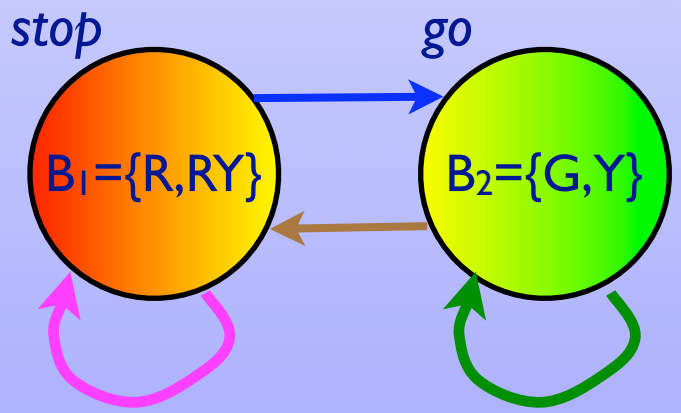


Language $\mathscr{L}$

$\varphi ::= stop \mid go \mid AXX\varphi$

No nontrivial strongly preserving abstract Kripke structure exists

The unique candidate abstract state space is:

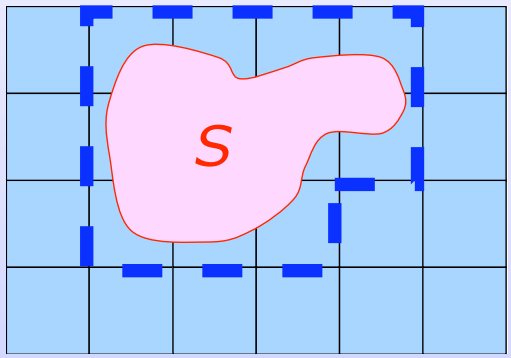


However, no abstract transition relation can be defined:

- It must be: $B_1 \rightarrow^{\#} B_2$ and $B_2 \rightarrow^{\#} B_1$
- This cannot be enough as $B_1 \not\models AXX\ go$
- If $B_1 \rightarrow^{\#} B_1$ then $B_1 \not\models AXX\ go$, contradiction
- Similarly for $B_2 \rightarrow^{\#} B_2$, contradiction

Strong Preservation by Abstract Interpretation

- State partitions are particular abstract domains



- Generic abstract domains instead of state partitions
- Correct abstract functions of temporal operators instead of abstract transition relations
- Strong preservation becomes (forward) completeness of abstract domains

Strong Preservation by Abstract Interpretation

- This allows to refine abstract models in order to make them strongly preserving
  - Generalized Paige-Tarjan-like algorithms for achieving strong preservation w.r.t. a whole language
  - Efficient algorithms for computing behavioural equivalences such as simulation equivalence
  - Abstract fixpoint-guided abstraction refinements instead of counterexample-guided abstraction refinements (CEGAR)

The Near Future

- The Italian AI community submitted a national research project proposal: AIDA - Abstract Interpretation Design and Applications. Some objectives:
  - Scalable and precise numerical abstract domains, abstract domains for points-to and aliasing analysis
  - Abstract interpretation of concurrent semantics based on event structures
  - Hiding software watermarks in loops and completeness holes
  - Transformations of temporal languages in order to get strong preservation
  - ...

Conclusion

- Italy lively contributed, contributes and will contribute to AI research
- A "stat": on DBLP, a search of papers with titles that include "AI" provides 431 hits: 37 authors have more than 5 papers in this list and 13 (35%) of them are Italians!

Italy thanks Patrick and Radhia
for inventing Abstract Interpretation!