

Basic advances in CMACS technology: Abstract Interpretation

Patrick Cousot

pcousot@cs.nyu.edu

<http://cs.nyu.edu/~pcousot>

CMU, Pittsburgh

November 3, 2011

Advances in abstract interpretation

Significant advances on

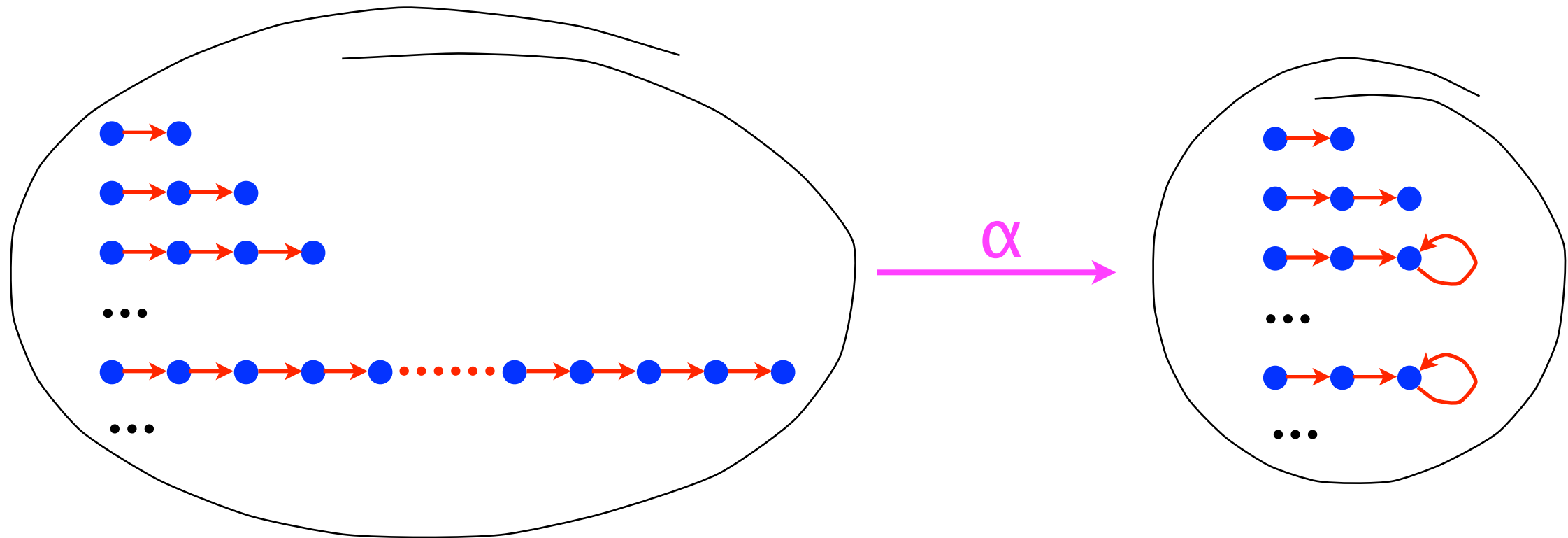
- Under-approximation
- Abstraction of unbounded array content
- Combination of algebraic and logical abstractions
- Probabilistic abstraction
- Termination/eventuality

have been done for infinite state systems.

Difficulty of the problems

- Abstraction to finite / bounded executions is impossible (unsound, ineffective, ...)

Example: [non]-termination of *unbounded* programs



- Abstraction must be infinite, which is extremely difficult

Under-approximation & arrays

- **Previously:** explore finite parts of a finite subset of executions
- **New:** algebraic approach to handle infinitely many infinite executions
- **Example:** pre-conditions ensuring the presence of errors

The screenshot displays the StaticChecker application window. The top pane shows the source code for `Max.cs` within the `RiSE.Tmp` project. The code defines a `namespace RiSE` containing a `public class Tmp` with a `public static void VMCAIPaperExample(string[] strings)` method. Inside this method, a `for` loop iterates over the `strings` array, asserting that each element is not null before setting it to null. The line `Contract.Assert(strings[i] != null);` is highlighted in blue.

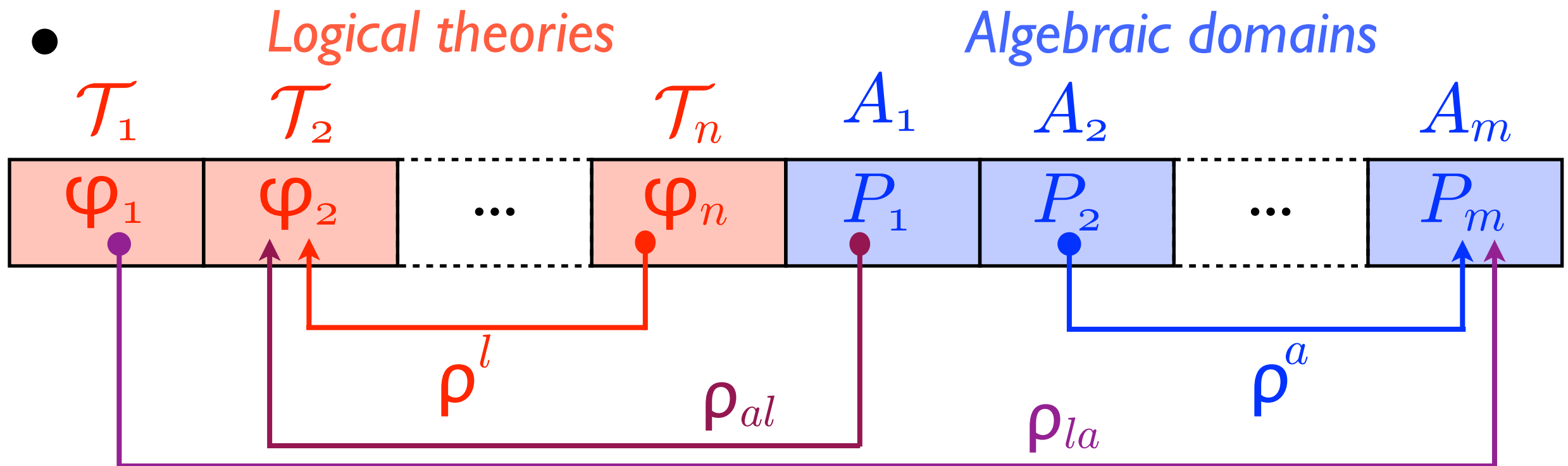
The bottom pane shows the `Error List` with 0 errors, 0 warnings, and 3 messages. The messages are as follows:

	Description	File	Line	Column	Project
1	CodeContracts: Suggested requires: <code>Contract.Requires(strings != null);</code>	Max.cs	11	12	StaticChecker
2	CodeContracts: Suggested precondition: <code>Contract.Requires(Contract.ForAll(0, strings.Length, i => strings[i] != null));</code>	Max.cs	11	12	StaticChecker
3	CodeContracts: Checked 10 assertions: 8 correct (2 masked)	Max.dll	1	1	StaticChecker

The bottom status bar includes tabs for `Error List`, `Output`, `Find Results 1`, `Find Symbol Results`, `Test Results`, and `Test Runs`.

Combining algebraic & logical abstractions

- A new understanding of the Nelson-Oppen procedure to combine logical theories in SMT solvers/provers as an algebraic reduced product



- When checking satisfiability of $\varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n$, the Nelson-Oppen procedure generates (dis)-equalities that can be propagated by ρ_{la} to reduce the $P_i, i=1, \dots, m$
- $\alpha_i(\varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n)$ can be propagated by ρ_{la} to reduce the $P_i, i=1, \dots, m$
- The purification to theory \mathcal{T}_i of $\gamma_i(P_i)$ can be propagated to φ_i by ρ_{al} in order to reduce it to $\varphi_i \wedge \gamma_i(P_i)$ (in \mathcal{T}_i)

Termination

- **Previously:** recent progress on automatic proof of termination for *small, simple and pure programs* (no abstraction needed)
- **Challenge:** scale automatic program termination methods to large, *complex, and realistic programs* by integrating *abstraction*
- **New advances:**
 - **Trace segments** as a new basis for inductively formulating program properties
 - **Fixpoint** definition of a **collecting semantics for termination/eventuality**
 - Systematic ways for constructing **termination proofs, by construction of abstract fixpoints** (e.g. variant functions)

Probabilistic abstraction

- Fixpoint concrete collecting semantics parameterized by probabilistic scenarios:
 - $\langle \Omega, \mathcal{E}, \mu \rangle$ **probabilistic space** (scenarios, observable events, probability measure)
 - $\langle \mathcal{D}, \leq \rangle$ conventional **semantic domain**
 - $S_p[[P]] \in \Omega \mapsto \mathcal{D}$ **probabilistic semantics**
 - $\wp(\Omega \mapsto \mathcal{D})$. concrete domain of **probabilistic properties**
- Several possible **abstractions**:
 - in the semantic domain
 - non-deterministic abstraction of the scenario domain
 - probabilistic abstraction in the scenario domainfor conditionals and loops
- Recover **classical probabilistic calculi and analyzes** by abstraction

Advances in aerospace applications

- The paper

Julien Bertrane, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, & Xavier Rival.

[Static Analysis and Verification of Aerospace Software by Abstract Interpretation](#). In *AIAA Infotech@Aerospace 2010*, Atlanta, Georgia. American Institute of Aeronautics and Astronautics, 20—22 April 2010. © AIAA.

received the **AIAA intelligent systems best paper award 2010**

- All **control/command software** of a European aircraft manufacturer now **mandatorily verified by abstract-interpretation based static analysis** (in conformance with DO-178-C)
- Progress on the static verification of **parallel processes**

The End