"The Reduced Product of Abstract Domains and the Combination of Decision Procedures" (1.) and "Termination: Foundations using Abstract Interpretation" (2.)

Patrick Cousot

pcousot@cs.nyu.edu http://cs.nyu.edu/~pcousot/

Joint ongoing work with

1. Radhia Cousot and Laurent Mauborgne
2. Radhia Cousot and Andreas Podelski

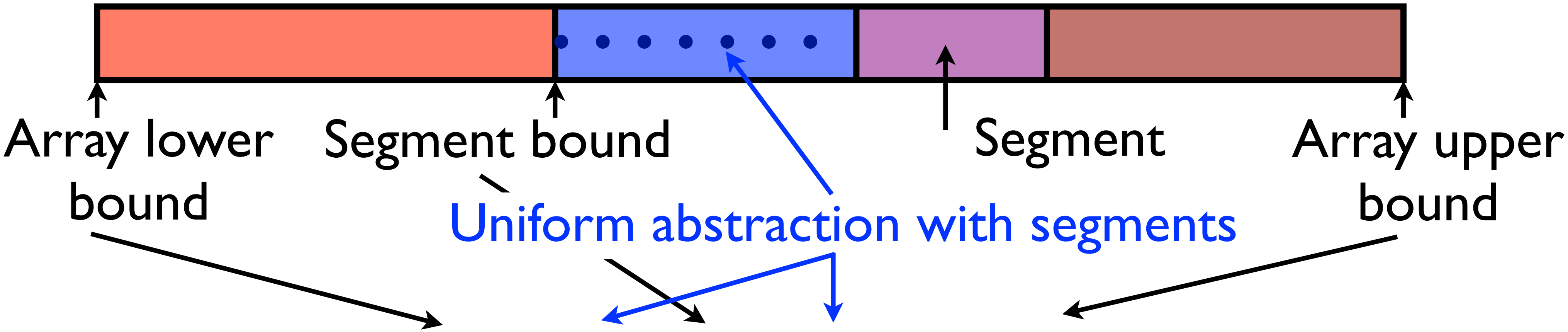
CMACS NYU meeting October 29, 2010

What has been done since Pittsburgh meeting

- Work since the Pittsburgh meeting:
  - Array content analysis (joint work with R. Cousot and F. Logozzo)
  - Segmented decision tree abstract domain (joint work with R. Cousot and L. Mauborgne)
  - Precondition inference from runtime-checked assertions (joint work with R. Cousot and F. Logozzo)
- Work in progress (today's presentations):
  - Probabilistic abstract interpretation: see talk by Michael Monerau
  - Logical abstract domains
  - Termination/Guarantee semantics, proof, and static analysis

Work on AI since the Pittsburgh meeting

Collection segmentation abstract domain

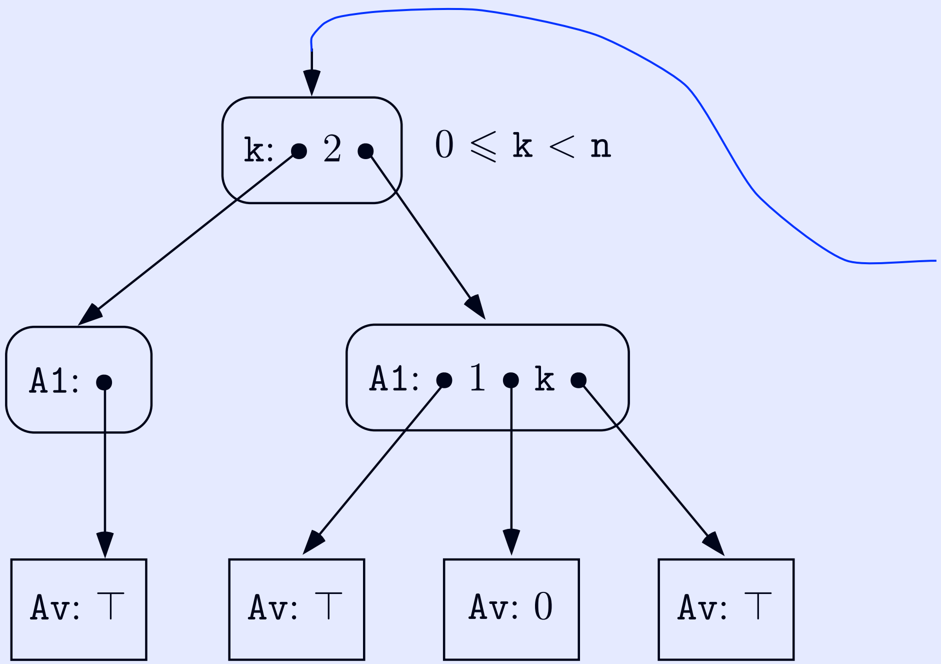


<{0},[0,0],{i}?,[2,+oo],{n}>

```
                                                   InitBackwards(int[] A) {
                                                     int i = A.Length;
                                         /* 1: */    while /* 2: */ (0 < i) {
                                         /* 3: */       i = i - 1;
                                         /* 4: */       A[i] = 0;
[ A: <{0 i} [0,0] {A.Length}> ]          /* 5: */     }
  [ i: [0,0] A.Length: [2,+oo] ]         /* 6: */  }
```

- Included by F. Logozzo in MSR Clousot (distributed with MS Visual Studio under Windows 7 pro)
- To appear in POPL'2011

Segmented decision tree abstract domain



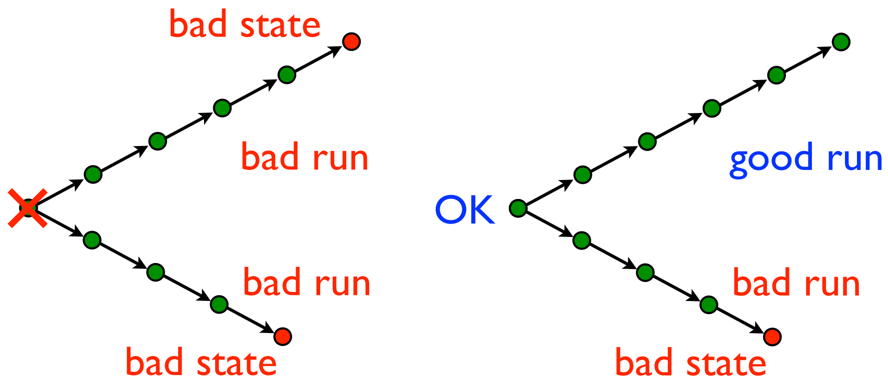

```
          int n; /* n > 0 */
          int k, A[n];
/* 0: */ k = 0;
/* 1: */  while /* 2: */ (k < n) {
/* 3: */      if (k > 0) {
/* 4: */         A[k] = 0;
/* 5: */       };
/* 6: */       k = k+1;
/* 7: */  };
/* 8: */
```

Decision tree

- Patrick Cousot, Radhia Cousot, Laurent Mauborgne: A Scalable Segmented Decision Tree Abstract Domain. Essays in Memory of Amir Pnueli, LNCS 6200, Springer, 2010: 72-95

Precondition inference from asserts

- Derive a static precondition from programmers and languages runtime-checked assertions in the code
- Not a wp:



- Symbolic under-approximation:

```
         void AllNotNull(Ptr[] A) {
/* 1: */  int i = 0;
/* 2: */  while /* 3: */
              (assert(A != null); i < A.length) {
/* 4: */   assert((A != null) && (A[i] != null));
/* 5: */   A[i].f = new Object();
/* 6: */   i++;
/* 7: */  }
/* 8: */ } 8: {0}𝔡{i,A.length}? - {0}𝔠{i,A.length}?
```

equal to/different from initial value

$\perp$, 𝔫, 𝔠. $\top$ [not]-check while unmodified

- To appear in VMCAI'2011

Ongoing work
(1) Logical abstract domains

Combining Algebraic and Logical Abstractions (I)

- Model-checking is "logical" (temporal logic, BDDs, SMT solvers,...)
- Abstract interpretation is "algebraic" (orders, lattices, linear algebra, categories, reduced product, ...)
- MC & AI can be combined within "set theory", e.g.
  - Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000, 12-25.
  - Patrick Cousot, Radhia Cousot: Refining Model Checking by Abstract Interpretation. Autom. Softw. Eng. 6(1): 69-95 (1999).

Combining Algebraic and Logical Abstractions (II)

- We propose a new MC & AI combination as "logical (i.e. SMT solvers) + algebraic (i.e. reduced product of abstract domains)"

Logical abstract domains: an instance of algebraic abstract domains

- Abstract properties: a theory (set of logical formulae)
  - Order =>, join: \/, meet: /\, ...
  - Concretization γ: interpretation
  - Abstraction α: in general does not exist (no best abstraction e.g. in absence of infinite conjunctions)
- Transformers:
  - Forward: Floyd/sp
  - Backward: Hoare/wp/wlp

$$(\mathtt{x} = 0) \Rightarrow (\mathtt{x} = 0 \vee \mathtt{x} = 1) \Rightarrow \ldots \Rightarrow \bigvee_{i=1}^{n} \mathtt{x} = i \Rightarrow \ldots$$

$$(\mathtt{x} \neq -1) \Leftarrow (\mathtt{x} \neq -1 \wedge \mathtt{x} \neq -2) \Leftarrow \ldots \Leftarrow \bigwedge_{i=1}^{n} \mathtt{x} \neq -i \Leftarrow \ldots$$

The (iterated) reduced product in AI

Reduced product

- A Cartesian product of abstract domains:

$$\Pi_{i=1,\ldots,n} A_i$$

- Understood as a conjunction:

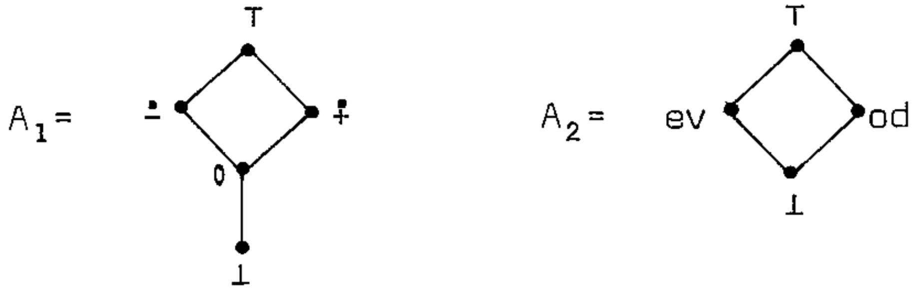
$$\gamma(a_1,\ldots,a_n) = \bigwedge_{i=1,\ldots,n} \gamma(a_i)$$

- Implemented as a Cartesian product plus a reduction ρ to propagate shared information from one component to another

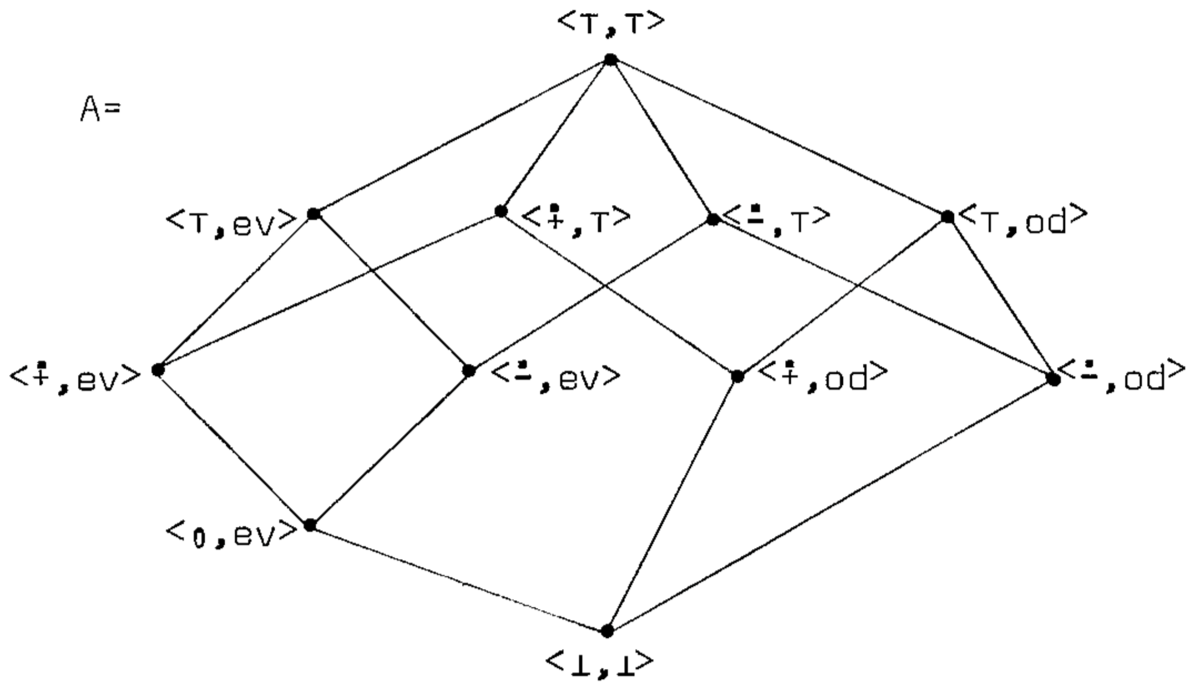
- Sound and complete

Example of reduced product

- Cousot & Cousot, POPL 79



reduced product:



Iterated pairwise reduction

- ρ_{ij} : reduction between abstract domains Ai and Aj
- $\vec{\rho}_{ij}$: extension to the Cartesian product

  $\vec{\rho}_{ij}(a_1,\ldots,a_n) = (a_1,\ldots, a'_i,\ldots, a'_j,\ldots, a_n)$

  where $(a'_i, a'_j) = \rho_{ij}(a_i, a_j)$
- Iterated pairwise reduction

  $\vec{\rho}^{\,*}$ = iterate the $\vec{\rho}_{ij}$, $i,j=1,\ldots,n$, $i\neq j$ until convergence
- Sound (but in general incomplete)

Example of pairwise iterated reduction

- Concrete domain: $L = \wp(\{a, b, c\})$
- Abstract domains: $A_1 = \{\emptyset, \{a\}, \top\}$
  $\top = \{a, b, c\}$ $A_2 = \{\emptyset, \{a, b\}, \top\}$
  $A_3 = \{\emptyset, \{a, c\}, \top\}$
- Reduction of $\langle \top, \{a, b\}, \{a, c\} \rangle$:
- Global: $\langle \{a\}, \{a, b\}, \{a, c\} \rangle$
- Pairwise reductions:
  $\vec{\rho}_{\vec{ij}}(\langle \top, \{a, b\}, \{a, c\} \rangle) = \langle \top, \{a, b\}, \{a, c\} \rangle$ for $\Delta = \{1, 2, 3\}, i, j \in \Delta, i \neq j$
- Iterated reduction:
  $\vec{\rho}^{\,*}(\langle \top, \{a, b\}, \{a, c\} \rangle) = \langle \top, \{a, b\}, \{a, c\} \rangle$

The Nelson-Oppen combination procedure

Objective of the Nelson-Oppen procedure

- Given deductive theories $\mathcal{T}_i$ in $\mathbb{F}(\Sigma_i)$, $\Sigma_i \dot{\subseteq} \Sigma$ with equality and decision procedures sat_i for satisfiability of quantifier free conjunctive formulæ $\varphi_i \in \mathbb{C}(\Sigma_i)$, $i = 1, \ldots, n$,
- Decide the satisfiability of a quantifier free conjunctive formula $\varphi \in \mathbb{C}(\bigcup_{i=1}^{n} \Sigma_i)$ in theory $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$ such that $\mathfrak{M}(\mathcal{T}) = \bigcap_{i=1}^{n} \mathfrak{M}(\mathcal{T}_i)$.

[23] G. Nelson and D. Oppen. Simplification by cooperating decision procedures. *TOPLAS*, 1(2):245–257, 1979.

The Nelson-Oppen combination procedure

1. Purification: project the quantifier-free conjunctive formula φ as an equi-satisfiable conjonction of component formulæ in each theory by introducing fresh variables for alien terms

$$\varphi' = \exists\, \vec{x}_1, \ldots, \vec{x}_n : \bigwedge_{i=1}^{n} \varphi_i \quad \text{where} \quad \varphi_i = \varphi'_i \wedge \bigwedge_{x_i \in \vec{x}_i} x_i = t_{x_i},$$

2. Repeat the equality reduction: propagate [dis] equalities deduced from each component formula φ_i to the other components formulæ) until no new [dis] equality can be added

3. Test satisfiability of component formulæ, unsatifiable iff one is unsatisfiable else unknown (originally, satisfiable if all component formula are satisfiable)

The Nelson-Oppen procedure is an iteratively reduced observation product

- The purification is a projection of the formula to an observation product (with auxiliary variables observing alien subterms)
- The reduction is iterative but only for [dis]equalities
- The unsatisfiability check is a reduction to (false)

Soundness of the procedure ?

- The unsatisfiability is sound
- More conditions for satisfiability soundness to ensure that all theories have isomorphic models such as
  - *stably-infinity*, *politeness*, etc ... so as to ensure that the models of the theories $\mathcal{T}_i$ have the same cardinalities
  - shared symbols (e.g. equality) have isomorphic interpretations in all theories sharing them or theories are disjoint which avoids the problem

Completeness of the procedure ?

- The procedure is incomplete

  so there exists formulæ satisfiable in two theories but not in their combination (e.g. integer arithmetics and bit vectors)

- Additional restrictions are necessary to ensure completeness

  - convexity (to avoid to have to reduce by disjunctions of [dis]equalities)

  - disjointness of the theories (but constants, to avoid to have to reduce on other properties than [dis] equality such as <)

Who cares about completeness in static analysis?

- We care about soundness but not on completeness (since we always get a sound overapproximation)
- Abandoning completeness, we can
  - combine theories sharing symbols other than = (as signs and parity)
  - perform reduction (even for non-convex theories) that are simply not optimal

Combining logical and algebraic abstractions

We use an iteratively reduced observation product with:

- logical components in logical abstract domains sharing symbols and handled by SMT solvers
- algebraic components in algebraic abstract domains
- the reduction propagates
  - [dis]equalities of logical components to all other components
  - pairwise algebraic reductions (equalities and others) to all other components

Perspectives

- A new perspective to combine
  - SMT solvers based model-checking understood as logical abstract domains (with logical widenings)
  - abstract interpretation-based static analysis using classical abstract domains (with algebraic widenings)
- This might avoid costly iterative refinement methods thanks to the expressivity of first-order logic

Ongoing work
(2) Termination

Basic idea:

Apply the abstract interpretation framework to termination

Abstract Interpretation framework

- Define the standard semantics: $\langle \Sigma, \tau \rangle$
- Define the collecting semantics (most general property of interest): $\mathcal{C} \in \wp(C)$
- Express the collecting semantics in fixpoint form:

$$\mathcal{C} = \mathsf{lfp}^{\subseteq} F \in \wp(C)$$

- Finite (MC) : compute $\mathcal{C}$ iteratively;
- Infinite (AI) : define an abstraction:

$$\langle \wp(C), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \sqsubseteq \rangle$$

Abstract Interpretation framework (cont'd)

- Define an abstract transformer:

$$\alpha \circ F \circ \gamma \sqsubseteq \overline{F}$$

- The fixpoint abstract semantics is sound:

$$\alpha(\mathsf{lfp}^{\subseteq} F) \sqsubseteq \mathsf{lfp}^{\sqsubseteq} \overline{F}$$

- Compute the abstract iterates iteratively:

$$\overline{F}^0 \triangleq \bot, \ldots, \overline{F}^{n+1} \triangleq \overline{F}(\overline{F}^n), \ldots$$

- Accelerating the convergence by widening ∇ and narrowing Δ (when necessary)

Termination analysis:

Applying the abstract interpretation framework to a termination collecting semantics

Standard semantics

- Traces on the set of states Σ :
  - Traces of length n: $\vec{s} = \vec{s}_0\vec{s}_1 \ldots \vec{s}_{n-1} \in \vec{\Sigma}^n$
  - Finite traces: $\vec{\Sigma}^+ \triangleq \bigcup_{n \geqslant 1} \vec{\Sigma}^n$
  - Infinite traces: $\vec{s} = \vec{s}_0\vec{s}_1 \ldots \vec{s}_i\vec{s}_{i+1} \ldots \in \vec{\Sigma}^\omega$
- Trace semantics: $\mathcal{S}_{\mathcal{T}} = \langle \Sigma, \text{init}, \text{final}, \vec{T} \rangle$
  - finite runs:

    $\forall n \geqslant 1 : \forall \vec{s} \in \vec{T} \cap \vec{\Sigma}^n : \vec{s}_0 \in \text{init} \wedge \forall i \in [0, n-1(: \vec{s}_i \notin \text{final} \wedge \vec{s}_{n-1} \in \text{final}$

  - Infinite runs:

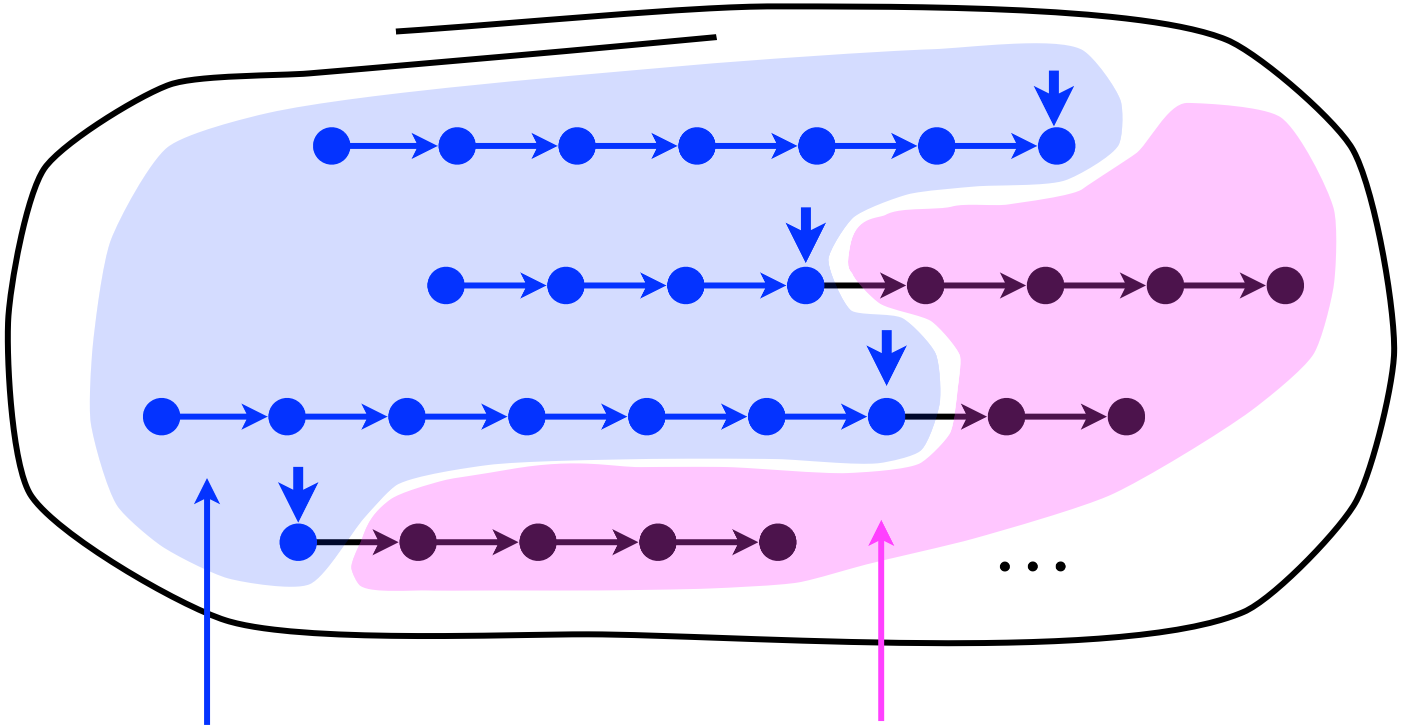
    $\forall \vec{s} \in \vec{T} \cap \vec{\Sigma}^\omega : \vec{s}_0 \in \text{init} \wedge \forall i \geqslant 0 : \vec{s}_i \notin \text{final}$

Example: traces generated by a transition system

- Transition system: $\langle \Sigma, \ \tau \rangle$
- Trace semantics: $\mathcal{S}_{\mathcal{T}}[\![\tau]\!] = \langle \Sigma, \text{ init}, \text{ final}, \vec{T} \rangle$
- Generated by the transition system:

$$\forall \vec{s}\, s s' \vec{s}\,' \in \vec{T} : \tau(s, s')$$

À la Floyd/Turing invariant/ranking function abstraction



Invariant x ranking function abstraction

Past/future abstraction

- **Past:**

$$\begin{array}{rcl} \alpha_{\leftarrow}(\vec{T}) & \triangleq & \{\vec{s} \in \vec{\Sigma}^{+} \mid \exists \vec{s}\,' \in \vec{\Sigma}^{\propto} : \vec{s}\,\vec{s}\,' \in \vec{T}\} \\ \mathcal{S}_{\leftarrow} & \triangleq & \langle \Sigma, \textsf{init}, \textsf{final}, \alpha_{\leftarrow}(\vec{T})\rangle \end{array}$$

- **Future:**

$$\begin{array}{rcl} \alpha_{\rightarrow}(\vec{T}) & \triangleq & \{\vec{s}\,' \in \vec{\Sigma}^{\infty} \mid \exists \vec{s} \in \vec{\Sigma}^{*} : \vec{s}\,\vec{s}\,' \in \vec{T}\} \\ \mathcal{S}_{\rightarrow} & \triangleq & \langle \Sigma, \textsf{init}, \textsf{final}, \alpha_{\rightarrow}(\vec{T})\rangle \end{array}$$

Past fixpoint semantics

- Past fixpoint semantics:

$$\begin{array}{rcl} \mathcal{B}_{\leftarrow}[\![\tau]\!] & \in & \wp(\vec{\Sigma}^{+}) \longrightarrow \wp(\vec{\Sigma}^{+}) \\ \mathcal{B}_{\leftarrow}[\![\tau]\!](\vec{X}) & \triangleq & \vec{\mathsf{init}}^{\,1} \cup \vec{X} \circ \vec{\tau} \\ \mathcal{S}_{\leftarrow}[\![\tau]\!] & = & \langle \Sigma,\ \mathsf{init},\ \mathsf{final},\ \mathsf{lfp}^{\subseteq}\, \mathcal{B}_{\leftarrow}[\![\tau]\!] \rangle \end{array}$$

- Further abstractions yield invariants (in fixpoint form):

$$\begin{array}{rcl} \alpha_i(\vec{T}) & \triangleq & \{\vec{s}_{n-1} \mid n \geqslant 1 \wedge \vec{s} \in \vec{T} \cap \vec{\Sigma}^{n}\} \\ \mathcal{S}_i & \triangleq & \langle \Sigma,\ \mathsf{init},\ \mathsf{final},\ \alpha_i \circ \alpha_{\leftarrow}(\vec{T}) \rangle \end{array}$$

- and automatic static analysis (iterative fixpoint computation with convergence acceleration by widening/narrowing)

Future fixpoint semantics

- Computational ordering

$$\vec{X} \mathrel{\vec{\sqsubseteq}} \vec{Y} \quad \triangleq \quad (\vec{X} \cap \vec{\Sigma}^{+}) \subseteq (\vec{Y} \cap \vec{\Sigma}^{+}) \wedge (\vec{X} \cap \vec{\Sigma}^{\omega}) \supseteq (\vec{Y} \cap \vec{\Sigma}^{\omega})$$

$$\langle \wp(\vec{\Sigma}^{\infty}),\ \vec{\sqsubseteq},\ \vec{\Sigma}^{\omega},\ \vec{\Sigma}^{+},\ \vec{\sqcup},\ \vec{\sqcap} \rangle \ \textit{is a complete lattice}$$

- Future fixpoint (termination collecting) semantics:

$$\mathcal{B}_{\rightarrow}[\![\tau]\!] \quad \in \quad \wp(\vec{\Sigma}^{\infty}) \longrightarrow \wp(\vec{\Sigma}^{\infty})$$

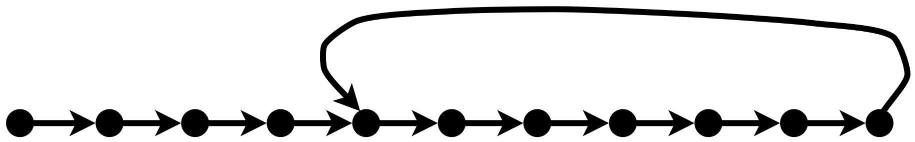
$$\mathcal{B}_{\rightarrow}[\![\tau]\!](\vec{X}) \quad \triangleq \quad \overrightarrow{\mathsf{final}}^{1} \cup (\vec{\tau} \circ \vec{X})$$

$$\mathcal{S}_{\rightarrow}[\![\tau]\!] \quad \triangleq \quad \langle \Sigma,\ \mathsf{init},\ \mathsf{final},\ \mathsf{lfp}^{\vec{\sqsubseteq}}\, \mathcal{B}_{\rightarrow}[\![\tau]\!] \rangle$$

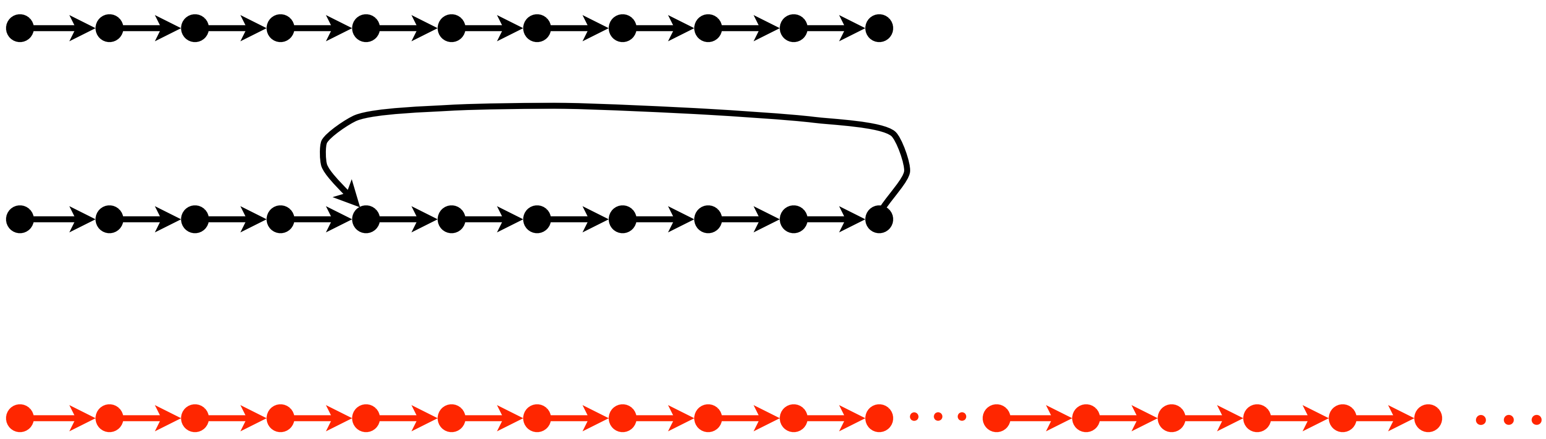
Patrick Cousot: Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. Theor. Comput. Sci. 277(1-2): 47-103 (2002)

Future of finite versus infinite systems

- Finite systems:



- Infinite systems:



Future approximation strategies

- Under-approximation of the termination domain:

$$\mathsf{dmn}[\neg(\vec{T} \cap \vec{\Sigma}^{\omega})]$$

$$\mathsf{dmn}[\vec{T}] \triangleq \{s \in S \mid \exists \vec{s} : s\vec{s} \in \vec{T}\}$$

  - Dual abstract interpretation (over-approximation of the complement)
  - Extremely difficult
  - Few known solutions (testing, bounded model-checking, symbolic execution, etc.), mostly ineffective

- Over-approximation of the termination argument:

  - Follow Lyapunov (stability), Turing, Floyd (ranking functions), Burstall, Ramsey, ...

The ranking abstraction

- Ordinals:

$$0 \triangleq \emptyset, 1 \triangleq \{0\}, 2 \triangleq \{0,1\}, \ldots, n+1 \triangleq \{0,\ldots,n\}, \ldots, \omega \triangleq \bigcup_{\delta<\omega} \delta, \omega+1, \ldots$$

- Ranking abstraction:

$$\begin{array}{rcl}\alpha_r(\vec{T}) & \triangleq & \{\langle \vec{s}_0,\ 0\rangle \mid \vec{s} \in \vec{T} \cap \vec{\Sigma}^1\} \\ & & \cup\, \{\langle s,\ \bigcup\limits_{ss'\vec{s} \in \vec{T} \wedge \langle s', \delta\rangle \in \alpha_r(\vec{T})} \delta + 1\rangle \mid \exists \vec{s}\,' \in \vec{\Sigma}^{\propto} : s\vec{s}\,' \in \vec{T}\} \\ \mathcal{S}_r & \triangleq & \langle \Sigma,\ \mathsf{init},\ \mathsf{final},\ \alpha_r \circ \alpha_{\rightarrow}(\vec{T})\rangle\end{array}$$

Ancestors abstraction

- Abstract a partial function by its domain of definition:

$$\alpha_a(f) \triangleq \mathsf{dmn}[f]$$

$$\mathcal{S}_a \triangleq \langle \Sigma, \mathsf{init}, \mathsf{final}, \alpha_a \circ \alpha_r \circ \alpha_{\rightarrow}(\vec{T}) \rangle$$

- We get **`pre[t*](final)`** [1]

(1) P. Cousot, Thesis, Grenoble, March 1978

Fixpoint ranking semantics

$$\begin{array}{rcl}
\mathcal{B}_r[\![\tau]\!] & \in & (\Sigma \not\longrightarrow \mathbb{O}) \longrightarrow (\Sigma \not\longrightarrow \mathbb{O}) \\
\mathcal{B}_r[\![\tau]\!](X) & \triangleq & \{\langle s,\ 0\rangle \mid s \in \mathsf{final}\} \\
& & \quad \cup \{\langle s, \displaystyle\bigcup_{\tau(s,s') \wedge \langle s',\, \delta\rangle \in X} \delta + 1\rangle \mid s \in \mathsf{pre}[\![\tau]\!](\mathsf{dmn}[X])\} \\
\mathcal{S}_r[\![\tau]\!] & = & \langle \Sigma,\ \mathsf{init},\ \mathsf{final},\ \mathsf{lfp}^{\subseteq}\, \mathcal{B}_r[\![\tau]\!]\rangle
\end{array}$$

PROOF

$$\alpha_r(\mathcal{B}_{\rightarrow}[\![\tau]\!](X))$$

$$= \ldots$$

$$= \mathcal{B}_r[\![\tau]\!](\alpha_r(X)) \qquad \text{(def. } \mathcal{B}_r[\![\tau]\!]\text{)}$$

Example

Consider the following program on $\mathbb{N}$.

```
while (i <> 1) {
   if even(i) { i = i div 2}
}
```

understood as defining the transition relation on $\mathbb{N}$

$$\tau(i, i') \quad \triangleq \quad i \neq 1 \wedge (odd(i) \Rightarrow i' = i) \wedge (even(i) \Rightarrow i' = i/2)$$

- Let us prove by fixpoint computation that the ranking semantics is:
  - Termination domain: $\mathsf{dom}[f] = \{2^n \mid n \in \mathbb{N}\}$
  - Ranking function: $f(n) = \log_2 n$

Iterates

we calculate the iterates of

$$\begin{aligned}\mathcal{B}_r[\![\tau]\!](f) &\triangleq \{\langle s,\ 0\rangle \mid s \in \mathsf{final}\} \cup \{\langle s,\ f(\tau(s)) + 1\rangle \mid \tau(s) \in \mathsf{dmn}[f]\}\\ &= \{\langle 1,\ 0\rangle\} \cup \{\langle i,\ f(i') + 1\rangle \mid i \neq 1 \wedge (odd(i) \Rightarrow i' = i) \wedge (even(i) \Rightarrow i' = i/2) \wedge \\ &\qquad i' \in \mathsf{dmn}[f]\}\end{aligned}$$

$$f^0 \triangleq \emptyset$$

$$f^1 \triangleq \mathcal{B}_r[\![\tau]\!](f^0) = \{\langle 1,\ 0\rangle\} \qquad \wr\text{since } \mathsf{dmn}[f^0] = \emptyset\wr$$

$$f^2 \triangleq \mathcal{B}_r[\![\tau]\!](f^1) = \{\langle 2,\ 1\rangle, \langle 1,\ 0\rangle\}$$

$$\wr\text{since } \mathsf{dmn}[f^0] = \{1\}\text{, and } \mathsf{pre}[\![\tau]\!](\mathsf{dmn}[f^0]) = \{2\} \text{ and } \tau(2,1)\wr$$

...

$$f^n = \{\langle 2^i,\ i\rangle \mid 0 \leqslant i < n\} \qquad \wr\text{induction hypothesis of the recurrence}\wr$$

$$\begin{aligned}f^{n+1} &\triangleq \mathcal{B}_r[\![\tau]\!](f^n) = \{\langle 1,\ 0\rangle\} \cup \{\langle 2^{i+1},\ i+1\rangle \mid 0 \leqslant i < n\}\\ &= \{\langle 2^i,\ i\rangle \mid 0 \leqslant i < n+1\}\end{aligned}$$

$\wr$since $\mathsf{dmn}[f^n] = \{2^i \mid 0 \leqslant i < n\}$, and $\mathsf{pre}[\![\tau]\!](\mathsf{dmn}[f^n]) = \{2^{i+1} \mid 0 \leqslant i < n\}$ and $\tau(2^{i+1}, 2^i)\wr$

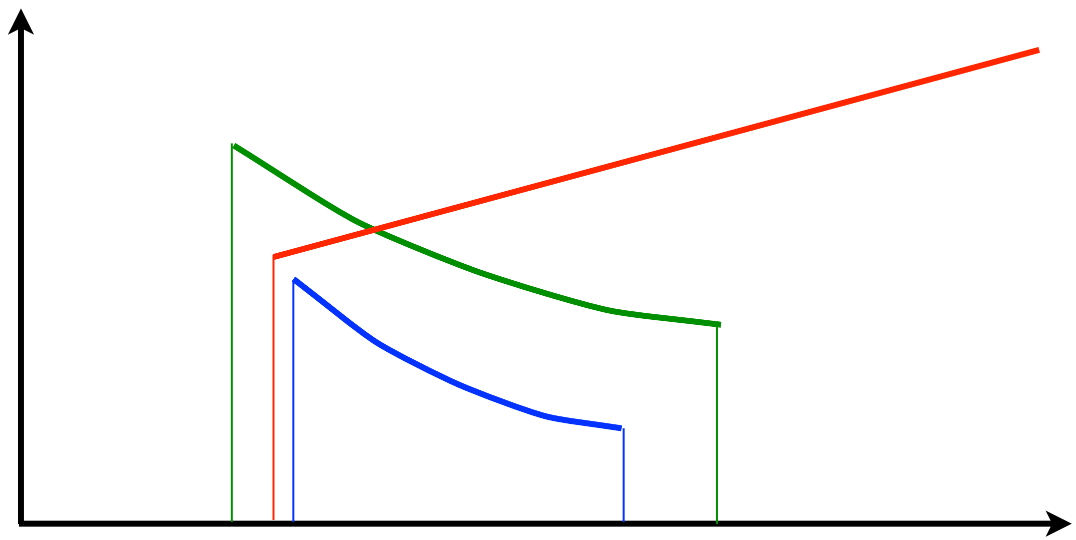
...

$$f^\omega = \dot{\bigcup_{n \geqslant 0}} f^n = \dot{\bigcup_{n \geqslant 0}} \{\langle 2^i,\ i\rangle \mid 0 \leqslant i \leqslant n\} = \{\langle 2^i,\ i\rangle \mid 0 \leqslant i\}$$

$$f^{\omega+1} = \mathcal{B}_r[\![\tau]\!](f^\omega) = f^\omega = \mathsf{lfp}_\emptyset^\subseteq \mathcal{B}_r[\![\tau]\!] = \lambda n \in 2^{\mathbb{N}} \bullet \log_2 n \qquad \square$$

Computable abstractions

- Approximation:



- Abstraction by a reduced product of standard abstractions e.g.:
  - Linear equalities (I) (with negative slopes and minimum or positive slopes and maximum)
  - Powers (II)
  - ...

(I) Michael Karr: Affine Relationships Among Variables of a Program. Acta Inf. 6: 133-151 (1976)
(II) Isabella Mastroeni: Algebraic Power Analysis by Abstract Interpretation. Higher-Order and Symbolic Computation 17(4): 297-345 (2004)

On going work ...

- Currently working on the formalization in AI terms
- and on abstractions for further methods:
  - Burstall (I), (II)
  - Ramsey (III)
  - ...
  - Checking temporal specifications of infinite systems (e.g. temporal logics)

(I) Rod M. Burstall: Program Proving as Hand Simulation with a Little Induction. IFIP Congress 1974: 308-312

(II) Patrick Cousot, Radhia Cousot: Sometime = Always + Recursion = Always on the Equivalence of the Intermittent and Invariant Assertions Methods for Proving Inevitability Properties of Programs. Acta Inf. 24(1): 1-31 (1987)

(III) Andreas Podelski, Andrey Rybalchenko: Transition Invariants. LICS 2004: 32-41

Conclusion

Conclusion

- This foundational preliminary work is the first step towards methods and inference algorithms for proving liveness by over-approximation