Extracting program logics from abstract interpretations defined by logical relations

David Schmidt

Kansas State University

`www.cis.ksu.edu/~schmidt`

From where do programming logics originate?

Consider Hennessy-Milner logic:

$c \models p$ is given, for primitive properties, p,

$c \models [f]\phi,$ if for all $c' \in f(c),\ c' \models \phi$

$c \models \langle f \rangle \phi,$ if there exists $c' \in f(c)$ such that $c' \models \phi$

for $c \in C$, where $f : C \to \mathcal{P}(C)$ denotes a nondeterministic transition function/action

Is "domain theory in logical form" [Abramsky02] hiding here?

We can deconstruct the logic to expose the underlying set-domains, for $S \subseteq C$:

$S \models \forall\phi,$ if for all $c \in S,\ c \models \phi$

$S \models \exists\phi,$ if there exists $c \in S$ such that $c \models \phi$

$c \models f;\phi,$ if $f(c) \models \phi$

(Read $[f]\phi$ as abbreviating $f;\forall\phi$, and read $\langle f \rangle \phi$ as abbreviating $f;\exists\phi$.)

The latter can be expanded into this logic, exposing lower- and upper-powerset constructions as well as function pre- and post-image:

$S \models_{L(\tau)} \forall(\bigvee_{i<k} \phi_i)$, if for all $c \in S$, there exists $j < k$ such that $c \models_\tau \phi_j$

$S \models_{U(\tau)} \bigwedge_{i<k}(\exists \phi_i)$, if for all $i < k$, there exists $c \in S$ such that $c \models_\tau \phi_i$

$c \models_{\tau_1} f; \phi$, if $f(c) \models_{\tau_2} \phi$, for $f : C_{\tau_1} \to C_{\tau_2}$

$f(c) \models_{\tau_2} f(\phi)$, if $c \models \phi_{\tau_1}$, for $f : C_{\tau_1} \to C_{\tau_2}$

This judgement set is extracted from Plotkin-style *logical relations* for the types, $\mathcal{P}_L(\tau)$, $\mathcal{P}_U(\tau)$, and $\tau_1 \to \tau_2$ [Plotkin80] , which generate a Cousot-Cousot-style *abstract interpretation* [Abramsky90, CousotCousot77] .

Preview of the talk

1. We show how to define an abstract interpretation via an approximation relation on base type, lifted to compound types via logical relations, *à la* Abramsky90.
2. We show the coincidence between Galois-connection-based approximation and relational approximation regarding functional soundness and *completeness*.
3. We show that every abstract domain has an *internal logic* and show how the logical relations generate logical operators within the internal logic.
4. When there are logical operators that do not fall within an abstract domain's internal logic, we show how to underapproximate them soundly by means of an *external logic* generated from the logical relations.

Abstract interpretation: computing on properties

```
readInt(x)
if x>0 :
   x:= pred(x)
x:= succ(x)
writeInt(x)
```

Q: Is output pos?

A: abstractly interpret domain Int by $Sign = \{neg, zero, pos, any\}$:

```
readSign(x)
if isPos(x):
   x:= pred♯(x)
x:= succ♯(x)
writeSign(x)
```

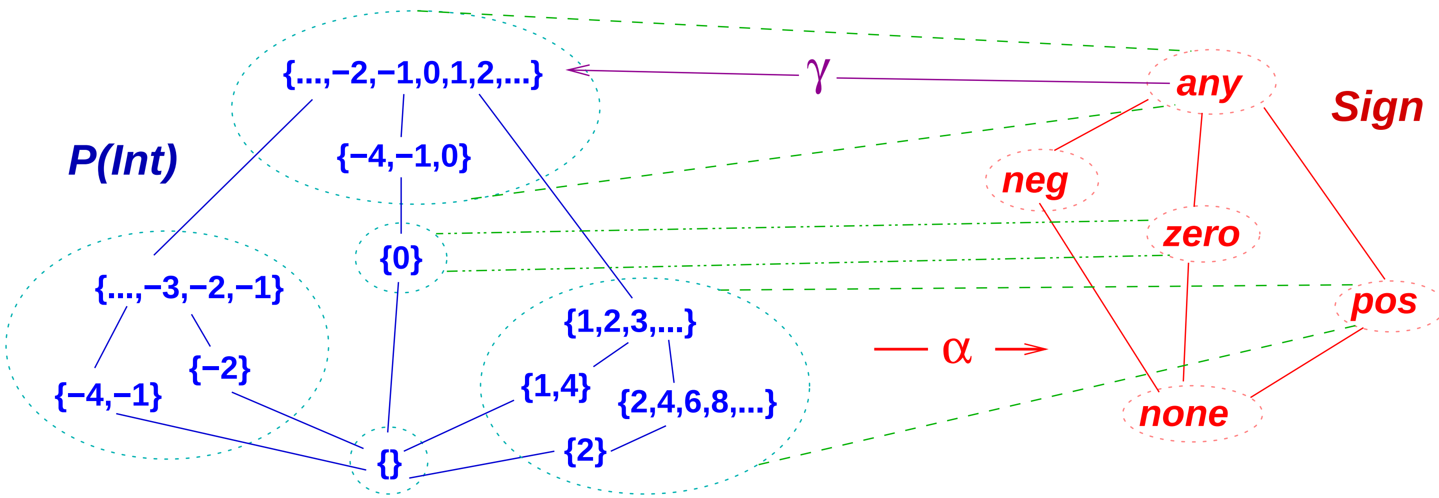
where
$$\begin{aligned} \mathrm{succ}^\sharp(pos) &= pos \\ \mathrm{succ}^\sharp(zero) &= pos \\ \mathrm{succ}^\sharp(neg) &= any \\ \mathrm{succ}^\sharp(any) &= any \end{aligned}$$
and
$$\begin{aligned} \mathrm{pred}^\sharp(neg) &= neg \\ \mathrm{pred}^\sharp(zero) &= neg \\ \mathrm{pred}^\sharp(pos) &= any \\ \mathrm{pred}^\sharp(any) &= any \end{aligned}$$

Calculate the *static analysis*:

$$\{zero \mapsto pos,\ neg \mapsto any,\ pos \mapsto any,\ any \mapsto any\}$$

The Question is decided only for $zero$ — the static analysis is *sound* but *incomplete*.

The Galois connection underlying the analysis



$\gamma : Sign \to \mathcal{P}(Int)$

$\gamma(none) = \{\}, \quad \gamma(any) = Int$

$\gamma(neg) = \{\cdots, -3, -2, -1\}$

$\gamma(zero) = \{0\}$

$\gamma(pos) = \{1, 2, 3, \cdots\}$

$\alpha : \mathcal{P}(Int) \to Sign$

$\alpha(S) = \sqcap\{a \mid \gamma(a) \subseteq S\}$

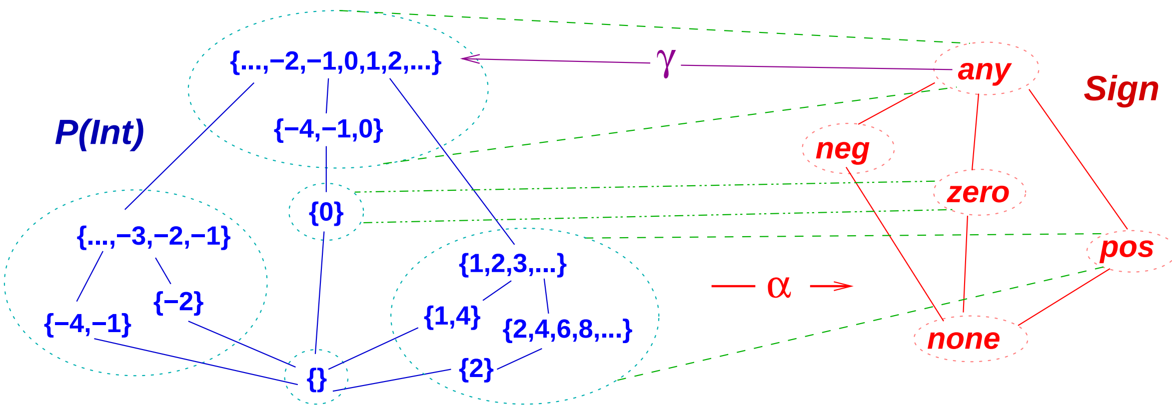
e.g., $\alpha\{2, 4, 6, 8, ...\} = pos$

$\alpha\{-4, -1, 0\} = any$

$\alpha\{0\} = zero$

$(\mathcal{P}(\mathrm{Int}), \subseteq)\langle\alpha, \gamma\rangle(Sign, \sqsubseteq)$ is a *Galois connection*: γ interprets the properties in $Sign$, and α maps each concrete set to the property that best describes the set [CousotCousot77] .

The Galois connection is a "completion" of an abstraction relation



For all $n > 0$, define $\rho_{Sign} \subseteq Int \times Sign$ as

$$-n \; \rho_{Sign} \; neg \qquad +n \; \rho_{Sign} \; pos$$

$$0 \; \rho_{Sign} \; zero \qquad m \; \rho_{Sign} \; any, \text{ for all } m \in Int$$

Example: $+3$ has property pos, because $+3 \; \rho_{Sign} \; pos$.

Intuition: for all $a \in Sign$, $\gamma(a) = \{i \in Int \mid i \; \rho_{Sign} \; a\}$.

$\rho \subseteq C \times A$ *generates a Galois connection between* C *and* A *iff* ρ *possesses* U-GLB-L-LUB-closure [Shmuely74,Schmidt04] .

A Galois connection defines an **internal logic** *that one uses to compute a static analysis*

For $(PC, \subseteq)\langle\alpha, \gamma\rangle(A, \sqsubseteq)$, for all $S \in PC$, and $a \in A$, define

$$S \models a \text{ iff } S \subseteq \gamma(a) \text{ (iff } \alpha(S) \sqsubseteq a\text{) (iff } S \;\bar{\rho}\; a\text{)}$$

Example: For $Sign$, $\{2, 5\} \models pos$. **The G.C. defines this *internal logic*:**

$$\phi ::= a \mid \phi_1 \sqcap \phi_2$$

because γ preserves $\sqcap$ as $\cap$ (that is, $\gamma(a_1 \sqcap a_2) = \gamma(a_1) \cap \gamma(a_2)$):

$$S \models a_1 \sqcap a_2 \text{ iff } S \models a_1 \text{ and } S \models a_2.$$

Example: In $Sign$, $\{2, 5\} \models pos \sqcap any$.

More importantly, for all $a \in A$, $a \sqsubseteq \phi$ implies $\gamma(a) \models \phi$.
Static analysis crucially depends on this (cf. the earlier example).

But $Sign$'s logic excludes disjunction, e.g., $\{0\} \models any = neg \sqcup pos$, yet $\{0\} \not\models neg$ and $\{0\} \not\models pos$ — γ does not preserve $\sqcup$ as $\cup$!

Abstract transformers compute on properties

For $f : PC \to PC$, $f^\sharp : A \to A$ is *sound* iff

$$\alpha \circ f \sqsubseteq f^\sharp \circ \alpha \quad \text{iff} \quad f \circ \gamma \sqsubseteq \gamma \circ f^\sharp$$

α and γ act as *semi-homomorphisms*; $f^\sharp$ is a *postcondition transformer*:

Example: For $succ : \mathcal{P}(Int) \to \mathcal{P}(Int)$, $succ\{0\} = \{1\}$, $succ^\sharp(zero) = pos$. *This is how a static analysis computes.*

Properties: $f(S) \models f^\sharp(\alpha(S))$ and $f(\gamma(a)) \models f^\sharp(a)$.

For example, $\{0\} \models zero$ and $succ\{0\} \models succ^\sharp(zero) = pos$.

$f^\sharp_{best} = \alpha \circ f \circ \gamma$ is the best — strongest postcondition — transformer in A's internal logic.

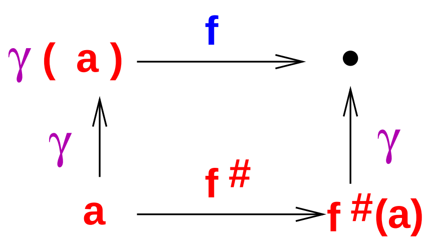
(Functional) completeness:
from semi-homomorphism to homomorphism

For $f : PC \rightarrow PC$, $f^\sharp : \mathcal{A} \rightarrow \mathcal{A}$:

Forwards(γ)-completeness

[GiacobazziQuintarelli01] :

$$f \circ \gamma = \gamma \circ f^\sharp$$



γ is a homomorphism from $\mathcal{A}$ to PC — it preserves $f^\sharp$ as f.

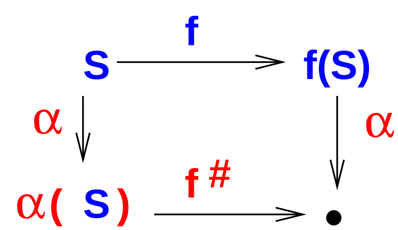
Theorem: $S \models f^\sharp(a)$ iff $S \subseteq f(\gamma(a))$.

That is, $f^\sharp$ **is a logical operator in $\mathcal{A}$'s internal logic (like $\sqcap$ is).**

Backwards(α)-completeness

[Cousots79,GiacobazziJACM00] :

$$\alpha \circ f = f^\sharp \circ \alpha$$



α is a homomorphism from PC to $\mathcal{A}$ — it preserves f as $f^\sharp$.

Theorem: $f^\sharp(\alpha(S)) \sqsubseteq a$ iff $f(S) \models a$.

That is, **we can *decide* properties of f in $\mathcal{A}$.**

Often, one wants **more** *than the internal logic*

$$\mathcal{L} \ni \phi ::= a \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid [f]\phi, \text{ where } a \in A$$

The interpretation, $[\![\cdot]\!] : \mathcal{L} \to \mathcal{P}(C)$, is defined as

$[\![a]\!] = \gamma(a)$

$[\![\phi_1 \wedge \phi_2]\!] = [\![\phi_1]\!] \cap [\![\phi_2]\!]$

$[\![\phi_1 \vee \phi_2]\!] = [\![\phi_1]\!] \cup [\![\phi_2]\!]$

$[\![[f]\phi]\!] = \widetilde{pre}_f[\![\phi]\!]$

where $\widetilde{pre}_f(S) = \{c \in C \mid f(c) \subseteq S\}$,

and $f : C \to \mathcal{P}(C)$ is a state-transition function

Define $S \models \phi$ iff $S \subseteq [\![\phi]\!]$. (In the *internal logic*, $S \models a$ iff $S \subseteq \gamma(a)$.)

$\phi_1 \vee \phi_2$ *and* $[f]\phi$ *might not fall in* A*'s internal logic.* (E.g., for $Sign$, there is no $\cup : Sign \times Sign \to Sign$ such that $\gamma(neg \cup pos) = [\![\phi_1 \vee \phi_2]\!]$. And, most $f^\sharp_{best}$ are not γ-complete for f.)

What justifies these extra operators? Can we employ them within a sound static analysis?

Help comes from the **logical relations** *for the types,* $\tau ::= b \mid \tau_1 \to \tau_2 \mid L(\tau) \mid U(\tau)$

For $\rho_\tau \subseteq C_\tau \times A_\tau$:

ρ_b is given, for base type b (e.g., $\rho_{Sign} \subseteq \mathrm{Int} \times Sign$)

$f \, \rho_{\tau_1 \to \tau_2} \, f^\sharp$ iff for all $c \in C_{\tau_1}$ and $a \in A_{\tau_1}$, $c \, \rho_{\tau_1} \, a$ implies $f(c) \, \rho_{\tau_2} \, f^\sharp(a)$

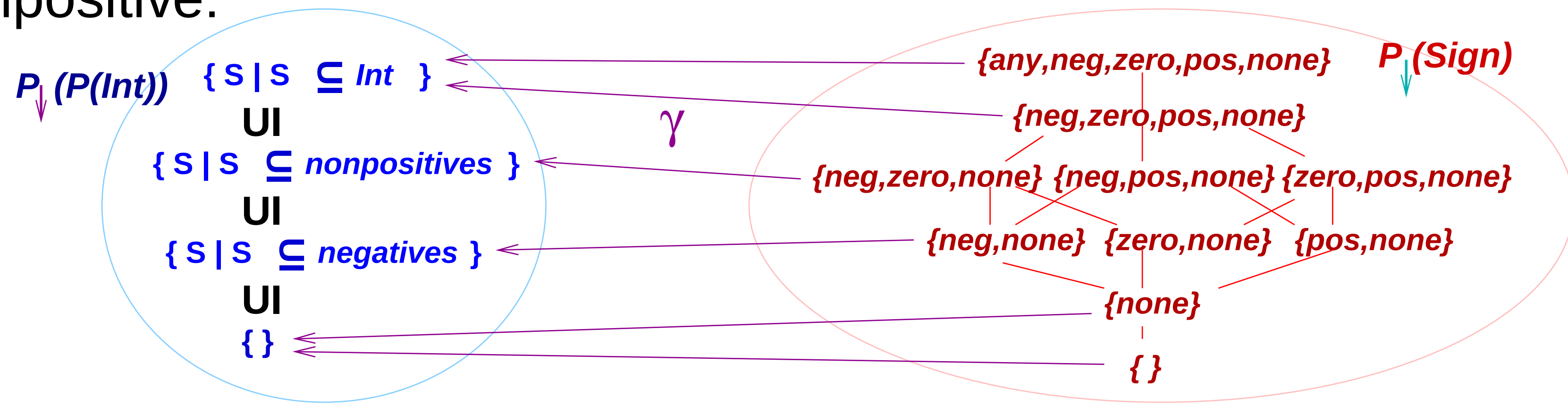
$S \, \rho_{L(\tau)} \, T$ iff for all $c \in S \in C_{L(\tau)}$, exists $a \in T \in A_{L(\tau)}$ s.t. $c \, \rho_\tau \, a$

$S \, \rho_{U(\tau)} \, T$ iff for all $a \in T \in A_{U(\tau)}$, exists $c \in S \in C_{U(\tau)}$ s.t. $c \, \rho_\tau \, a$

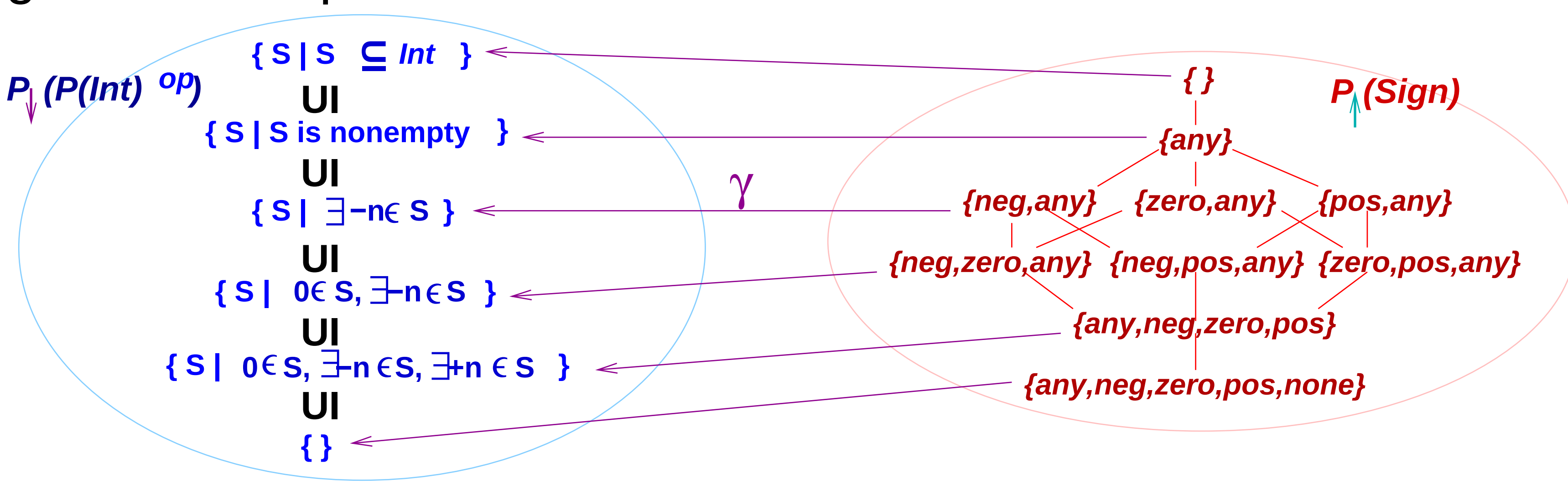
where

- D_b is given (e.g., *Int* and *Sign*)
- $D_{\tau_1 \to \tau_2} = D_{\tau_1} \to D_{\tau_2}$, the poset of monotone functions from D_{τ_1} to D_{τ_2}.
- $D_{L(\tau)} = \mathcal{P}_L(D_\tau)$, a *lower powerset* of D_τ, a collection of downclosed subsets of D that includes all $\downarrow d$ for all $d \in D$, partially ordered by $\subseteq$, and closed under $\cap$.
- $D_{U(\tau)} = \mathcal{P}_U(D_\tau)$, an *upper powerset* of D_τ, a collection of upclosed subsets of D that includes all $\uparrow d$ for all $d \in D$, partially ordered by $\supseteq$, and closed under $\cup$.

Lower-powerset approximation defines universal, disjunctive properties: e.g., $\{neg, zero, none\}$ asserts $\forall(neg \vee zero)$ — all data are nonpositive:



Upper-powerset approximation defines conjunctive, existential properties: e.g., $\{neg, pos, any\}$ asserts $\exists neg \wedge \exists pos$ — there exists a negative and a positive datum:



Consequences of the "powerset lift"

- When ϕ_i are found in A's internal logic, then $\forall(\bigvee_i \phi_i)$ is found in $\mathcal{P}_\downarrow(A)$'s internal logic.
- When ϕ_i are found in A's internal logic, then $\bigwedge_i(\exists\phi_i)$ is found in $\mathcal{P}_\uparrow(A)$'s internal logic.

More importantly, we use down-closed and up-closed subsets of A to define an *external logic* where judgements take form,

$$a \in [\![\phi]\!]^A \subseteq A, \quad \text{rather than} \quad a \sqsubseteq \phi \in A.$$

The sets let us define sound *underapproximation*, where $a \in [\![\phi]\!]^A$ implies $\gamma(a) \subseteq [\![\phi]\!]$.

The external logic falls even further outside of A's internal logic because of problems with $f^\sharp : A_1 \to A_2$, which we "split" into pre- and post-image, which are rarely γ-complete....

A programming logic based on logical relations

Types: $\tau ::= b \mid L(\tau) \mid U(\tau) \mid \tau_1 \to \tau_2$

Assertions: $\phi ::= a \mid \bigvee_{i<k} \phi_i \mid \bigwedge_{i<k} \phi_i \mid f(\phi) \mid f;\phi$

Judgement typing:

$$a : b \qquad \frac{\phi_i : \tau, \text{ for all } i < k}{\bigvee_{i<k} \phi_i : L(\tau)} \qquad \frac{\phi_i : \tau, \text{ for all } i < k}{\bigwedge_{i<k} \phi_i : U(\tau)}$$

$$\frac{f : \tau_1 \to \tau_2 \quad \phi : \tau_1}{f(\phi) : \tau_2} \qquad \frac{f : \tau_1 \to \tau_2 \quad \phi : \tau_2}{f;\phi : \tau_1}$$

Concrete judgements: have form, $c \models_\tau \phi$, where $c \in C_\tau$ and $\phi : \tau$

$c \models_b a$ is given by $\rho_b \subseteq C_b \times A_b$, e.g., $n \models_{Sign} a$ if $n \, \rho_{Sign} \, a$

$S \models_{L(\tau)} \bigvee_{i<k} \phi_i$, if for all $c \in S$, there exists $j < k$ such that $c \models_\tau \phi_j$

$S \models_{U(\tau)} \bigwedge_{i<k} \phi_i$, if for all $i < k$, there exists $c \in S$ such that $c \models_\tau \phi_i$

$c \models_{\tau_1} f;\phi$, if $f(c) \models_{\tau_2} \phi$, for $f \in C_{\tau_1} \to C_{\tau_2}$ (this defines $c \in \widetilde{pre}_f(\phi)$)

$c \models_{\tau_2} f(\phi)$, if there exists $c' \in C_{\tau_1}$ such that $c' \models_{\tau_1} \phi$ and $f(c') = c$,

for $f \in C_{\tau_1} \to C_{\tau_2}$ (this defines $c \in post_f(\phi)$)

The corresponding external logic for abstract domains

Abstract judgements have form, $a \models^{\mathcal{A}}_{\tau} \phi$,

where $a \in A_\tau$ and $\phi : \tau$. **(Read $a \models^{\mathcal{A}}_{\tau} \phi$ as $a \in [\![\phi]\!]^{A}_{\tau}$.)**

$a \models^{\mathcal{A}}_{b} a'$, if $a \sqsubseteq_b a'$, for $a, a' \in A_b$ (e.g., $pos \sqsubseteq_{Sign} any$)

$T \models^{\mathcal{A}}_{L(\tau)} \bigvee_{i<k} \phi_i$, if for all $a \in T$, there exists $j < k$ such that $a \models^{\mathcal{A}}_{\tau} \phi_j$

$T \models^{\mathcal{A}}_{U(\tau)} \bigwedge_{i<k} \phi_i$, if for all $i < k$, there exists $a \in T$ such that $a \models^{\mathcal{A}}_{\tau} \phi_i$

$a \models^{\mathcal{A}}_{\tau_1} f; \phi$, if $f^\sharp(a) \models^{\mathcal{A}}_{\tau_2} \phi$, where $f \, \rho_{\tau_1 \to L(\tau_2)} \, f^\sharp$

(this underapproximates $\widetilde{pre}_f(\phi)$)

$a \models^{\mathcal{A}}_{\tau_2} f(\phi)$, if there exists $a' \in A_{\tau_1}$ such that $a' \models^{\mathcal{A}}_{\tau_1} \phi$

and $a' \in f^\flat(a)$, where $f^{-1} \, \rho_{\tau_1 \to U(\tau_2)} \, f^\flat$

(this underapproximates $pre_{f^{-1}}(\phi) = post_f(\phi)$)

Consequences

1. ***Soundness***: $S \; \rho_\tau \; a$ and $a \models^{\mathcal{A}}_\tau \phi$ imply $S \models_\tau \phi$.
2. ***Completeness I***: When a logical operator, $f^\sharp$, is γ*-complete* for f, then the judgement form, $\cdot \models^{\mathcal{A}}_\tau f^\sharp(\phi)$, is complete (falls in the internal logic) for $\cdot \models_\tau f(\phi)$.
3. ***Completeness II***: When $f^\sharp$ is α-complete for f, then $\cdot \models^{\mathcal{A}}_\tau f^\sharp; \phi$ is complete for $\cdot \models_\tau f; \phi$
4. We can formally justify branching-time logics (e.g., Hennessy-Milner logic) as sound, best approximating, and complete for static analysis (*abstract model checking* [Dams97]).

References **This talk:** www.cis.ksu.edu/~schmidt/papers

1. S. Abramsky. Abstract interpretation, logical relations, and Kan extensions. *J. Logic and Comp.* (1)1990.
2. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. ACM POPL 1979.
3. D. Dams, et al. Abstract interpretation of reactive systems. *ACM TOPLAS* (19)1997.
4. R. Giacobazzi and E. Quintarelli. Incompleteness, counterexamples, and refinements in abstract model checking. SAS'01, LNCS 2126.
5. G. Plotkin. *Domains.* Course lecture notes, Pisa/Edinburgh, 1984.
6. G. Plotkin. Lambda-definability in the full type hierarchy. In *To H.B. Curry*, Academic Press, 1980.
7. D.A. Schmidt. A calculus of logical relations for over- and underapproximating static analyses. *Sci. Comp. Prog.* (64)2007.