institute IMDEA software

Complete Abstraction on Bounded Domains

BACKGROUND

START

Abstract Interpretation?

...an example

```
if x < 0:
  x <- -2x
else:
  x <- x+1
return x
```

Can the output be negative?

INTUITIVELY:

Group possible inputs by abstracting on relevant properties only

x negative:

```
if <0 < 0:
  >0 <- -2<0
else:
  x <- x+1
return >0
```

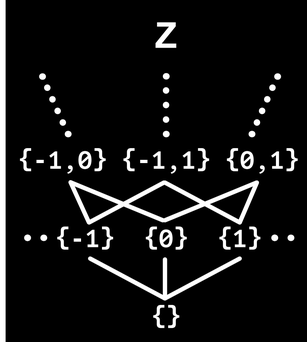
x positive: (or zero)

```
if ≥0 < 0:
  x <- -2x
else:
  >0 <- ≥0 +1
return >0
```

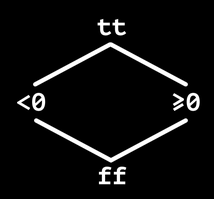
more formally...

DOMAINS:

Concrete



Abstract



POSSIBLY INFINITE

parametrized by

Execute an *abstract* approximation of the *concrete* semantics

THE IDEAL SCENARIO: A COMPLETE ABSTRACTION

Executing the abstract semantics = Abstracting the concrete output

!!!

(length of the longest chain)

If the HEIGHT of a domain is ≤ than some constant *k* the domain is BOUNDED

*

OUR CONTRIBUTION:

THE IDEA

The *"tricky"* parts of the semantics are the fixpoint computations

Kleene Fixpoint Th.

Computing fixpoints is done by following an *ascending chain* in the domain

...but! all chains in a bounded domain are "short"*

⇒ analyzing properties on the abstract semantics becomes easy

e.g. *we can decide program termination of the abstract semantics*
(hint: we can remove while loops)

\+ Completeness allows us to move our analysis back to the *concrete* semantics.

THE RESULT *(more in the full paper)*

If a program admits a complete abstraction on a bounded domain

then

we can decide its termination

!!!

...and so?

Allows us to reason about the power and tradeoffs of complete abstractions.

Establishes a parallel between expressivity classes and domain topologies.

for details, and more

Full Paper?

doi.org/10.1007/978-3-031-22308-2_9

R. Bruni, R. Gori, N. Manini (Static Analysis Symposium, SAS 2022).
"Deciding Program Properties via Complete Abstractions on Bounded Domains".