Abstract Interpretation

- **Abstraction Interpretation** is a general theory that formalizes the notion of approximation.
- “A program denotes computations in some universe of objects. Abstract interpretation of programs consists in using that denotation to describe computations in another universe of abstract objects, so that the results of abstract execution give some information on the actual computations” — P. Cousot

- Is \( 1/(x+1+y) \) well defined at run-time when used in a given program \( P \), assuming any possible input?

**Concrete Semantics**

- **WELL DEFINED**

**Abstract Semantics 1**

- **UNKNOWN**

**Abstract Semantics 2**

- **WELL DEFINED**

---

**Example Abstractions**

### Signs

\[
\begin{align*}
\{ x \geq 0 \\
y \geq 0 \}
\end{align*}
\]

### Intervals

\[
\begin{align*}
\{ x \in [19, 77] \\
y \in [20, 07] \}
\end{align*}
\]

### Octagons

\[
\begin{align*}
1 \leq x \leq 9 \\
y \leq 5 \\
1 \leq y \leq 5 \\
y \leq y \leq 0
\end{align*}
\]

### Polyhedra

\[
\begin{align*}
19x + 77y \leq 207 \\
20x + 07y \geq 0
\end{align*}
\]

---

**Example Definitions**

// Simple sign abstraction for concrete values

```c
typedef
// All possible values in this lattice
enum values {Bot, Neg, Zer, Pos, Inf, Err, Top};
// Printable representation for the output
friend std::ostream & operator<*(std::ostream &os, const sign);
// Default constructor
sign(values = Bot);
// Lattice operations
static bool eq(const sign, const sign);
static bool leq(const sign, const sign);
static bool leq(const sign, const sign);
// Abstract function
template<typename T>
static sign alpha(T t);
// Current lattice value
values value;
```

---

**Why bother?**

- **Ariane 5 Explosion**
  - **Cause**: narrowing conversion
  - **Cost**: $500 million

- **NASA Mars Climate Orbiter**
  - **Cause**: mixing metric & imperial units
  - **Cost**: $125 million

- **LA Air-Traffic Controller**
  - **Cause**: counter underflow
  - **Cost**: unknown

- **Patriot Missile Failure**
  - **Cause**: finite precision rounding
  - **Cost**: 28 dead, 100 injured

- **Zune's New Year Freeze**
  - **Cause**: infinite loop
  - **Cost**: New Year's Eve without music

- **Pentium FDIV bug**
  - **Cause**: incomplete entries in a look-up-table
  - **Cost**: $400 million

---

**Abstract Interpretation Library**

- **The model is different**
  - instead of analyzing we (abstractly) evaluate
- The model closely resembles C++ itself
- **Incremental refinements**
  - there are defaults for lack of any information
  - we can trade precision for speed or trying sooner
- Plenty of developed abstract domains
  - existing libraries to support them in a given language
- Abstract domains compose & cooperate
  - reduced product
- No need to work with AST of a program
  - similar to OO libraries
- **Application specific abstract domains**
  - C++ entities in the analyzed program can be used in analyzer

---

**Abstract Interpretation can be used to detect or prove absence of these bugs!**