Goal of abstraction is to reduce complexity.

For example:

- make infinite-state program finite-state (enables model checking, say of programs with \texttt{int}'s)
- make large program small

In practice: we omit detail from the program.
Example: Abstraction

Consider computation of $p = 2^e$:

Concrete:

```c
unsigned int e;
int p = 1;
while (e > 0) {
    p = 2 * p;
    --e;
}
```

$G(2 \mid p)$

$(2 \mid p) \implies G(2 \mid p)$
Example: Abstraction

Consider computation of $p = 2^e$:

**Concrete:**
```c
unsigned int e;
int p = 1;
while (e > 0) {
    p = 2 * p;
    --e;
}
```

**Abstract:**
```c
unsigned int e;
bool pdiv2 = false;
while (e > 0) {
    pdiv2 = true;
    --e;
}
```

$G(2 | p)$

$(2 | p) \Rightarrow G(2 | p)$

$G pdiv2$

$pdiv2 \Rightarrow G pdiv2$
Critical point: relationship between

**concrete** (original) program and **abstract** (simplified) program.

An abstraction can be:

(a) **inexact**: abstract pgm. satisfies different properties

(b) **exact**: abstract pgm. satisfies same properties
    (although it is smaller!)
Critical point: relationship between

- concrete (original) program and
- abstract (simplified) program.

An abstraction can be:

(a) **inexact**: abstract pgm. satisfies different properties
(b) **exact**: abstract pgm. satisfies same properties
   (although it is smaller!)

Is (a) useful? Is (b) useful? Is it possible?
Consider program $\mathcal{P}$, abstract program $\mathcal{P}'$, a class $\mathcal{C}$ of properties of interest, and:

\[
\forall \phi \in \mathcal{C} : \quad \mathcal{P}' \models \phi \Rightarrow \mathcal{P} \models \phi \tag{1}
\]

\[
\forall \phi \in \mathcal{C} : \quad \mathcal{P} \models \phi \Rightarrow \mathcal{P}' \models \phi \tag{2}
\]

“exact”: both (1) and (2) hold

“inexact”: we expect either (1) or (2) to hold
Consider program $\mathcal{P}$, abstract program $\mathcal{P}'$, a class $\mathfrak{c}$ of properties of interest, and:

$$\forall \phi \in \mathfrak{c} : \mathcal{P}' \models \phi \Rightarrow \mathcal{P} \models \phi$$

(1)

$$\forall \phi \in \mathfrak{c} : \mathcal{P} \models \phi \Rightarrow \mathcal{P}' \models \phi$$

(2)

“exact”: both (1) and (2) hold

“inexact”: we expect either (1) or (2) to hold

If neither (1) nor (2) hold, the abstraction is usually meaningless. (Why?)
How to Work with Abstractions

**Goal:** prove properties of $\mathcal{P}$ by verifying (the smaller) $\mathcal{P}'$.

If $\mathcal{P}'$ is exact abstraction of $\mathcal{P}$:

- Verify $\phi$ over $\mathcal{P}'$, and report the result.
How to Work with Abstractions

**Goal:** prove properties of $\mathcal{P}$ by verifying (the smaller) $\mathcal{P}'$.

If $\mathcal{P}'$ is **exact** abstraction of $\mathcal{P}$:

- Verify $\phi$ over $\mathcal{P}'$, and report the result.

If $\mathcal{P}'$ is **inexact** and is **conservative**, i.e. satisfies

$$\forall \phi \in C : \mathcal{P}' \models \phi \Rightarrow \mathcal{P} \models \phi : \tag{1}$$

- Verify $\phi$ over $\mathcal{P}'$. If it holds, we are done.

Using (1), we can **verify**, but not falsify!

Using (2) (previous slide), we can **falsify**, but not verify.