Static Analysis for Data Science

MPRI 2-6: Abstract Interpretation, Application to Verification and Static Analysis

Caterina Urban
February 7th, 2022
Year 2021-2022
Data Science is Everywhere

data is cheap and ubiquitous

- web logs
- mobile devices
- sensors
- transactions

Data science is revolutionizing industries

- retail
  - personalized recommendations
  - targeted marketing
- pharmaceutical
  - predictive models
  - patient selection
- manufacturing
  - equipment failure predictions
  - internet of things
- finance
  - predictive models
  - customized product offerings
- energy
  - exploration and discovery
  - accident prevention
- health care
  - personalized treatments
  - preventive care
data → pre-processing → model training → model deployment → prediction

data science
software

today → next week
Ubiquitous Programming Errors

data science means *programming*

*programming* means *programming errors*

programming errors that do not cause failures can have **serious consequences**

- pharmaceutical
- finance
- health care
- energy
Anomalously Unused Data
In this paper, we exploit a new multi-country historical dataset on public (government) debt to search for a systemic relationship between high public debt levels, growth and inflation. Our main result is that whereas the link between growth and debt seems relatively weak at “normal” debt levels, median growth rates for countries with public debt over roughly 90 percent of GDP are about one percent lower than otherwise; average (mean) growth rates are several percent lower. Surprisingly, the relationship between public debt and growth is remarkably similar across emerging markets and advanced economies. This is not the case for inflation. We find no systematic relationship between high debt levels and inflation for advanced economies as a group (albeit with individual country exceptions including the United States). By contrast, in emerging market countries, high public debt levels coincide with higher inflation.

Our topic would seem to be a timely one. Public debt has been soaring in the wake of the recent global financial crisis, especially in the epicenter countries. This should not be surprising, given the experience of earlier severe financial crises. Outsize deficits and epic bank bailouts may be useful in fighting a downturn, but what is the long-run macroeconomic impact, especially against the backdrop of graying populations and rising social insurance costs? Are sharply elevated public debts ultimately a manageable policy challenge?

Our approach here is decidedly empirical, taking advantage of a broad new historical dataset on public debt (in particular, central government debt) first presented in Carmen M. Reinhart and Kenneth S. Rogoff (2008, 2009b). Prior to this dataset, it was exceedingly difficult to get more than two or three decades of public debt data even for many rich countries, and virtually impossible for most emerging markets. Our results incorporate data on 44 countries spanning about 200 years. Taken together, the data incorporate over 3,700 annual observations covering a wide range of political systems, institutions, exchange rate and monetary arrangements, and historic circumstances.

We also employ more recent data on external debt, including debt owed both by governments and by private entities. For emerging markets, we find that there exists a significantly more severe threshold for total gross external debt (public and private)—which is almost exclusively denominated in a foreign currency—than for total public debt (the domestically issued component of which is largely denominated in home currency). When gross external debt reaches 60 percent of GDP, annual growth declines by about two percent; for levels of external debt in excess of 90 percent of GDP, growth rates are roughly cut in half. We are not in a position to calculate separate total external debt thresholds (as opposed to public debt thresholds) for advanced countries. The available time-series is too recent, beginning only in 2000. We do note, however, that external debt levels in advanced countries now average nearly 200 percent of GDP, with external debt levels being particularly high across Europe.

The focus of this paper is on the longer term macroeconomic implications of much higher public and external debt. The final section, how-
The Reinhart-Rogoff

Growth in a Time of Debt

By Carmen M. Reinhart and Kenneth S. Rogoff

In this paper, we exploit a new multi-country historical dataset on public and external debt to search for a systemic relationship between high public debt levels, growth, and inflation. Our main result is that whereas the link between growth and debt seems relatively weak at "normal" debt levels, median growth rates for countries with public debt over roughly 90 percent of GDP are about one percent lower than otherwise; average (mean) growth rates are several percent lower. Surprisingly, the relationship between public debt and growth is remarkably similar across emerging markets and advanced economies. This is not the case for inflation. We find no systematic relationship between high debt levels and inflation for advanced economies as a group (albeit with individual country exceptions including the United States). By contrast, in emerging market countries, high public debt levels coincide with higher inflation.

Our topic would seem to be a timely one. Outed deficits and epic bank bailouts may be useful in fighting a downturn, but what is the long-run macroeconomic impact, especially against the backdrop of a greying population and rising social insurance costs? Are sharply elevated public debts ultimately a manageably policy challenge?

Our approach here is decidedly empirical, taking advantage of a broad new historical dataset on public debt (in particular, central government debt) first presented in Carmen M. Reinhart and Kenneth S. Rogoff (2008, 2009). Prior to this dataset, it was exceedingly difficult to get more than two or three decades of public debt data even for many rich countries, and virtually impossible for most emerging markets. Our results incorporate data on 44 countries spanning about 200 years. Taken together, the data incorporate over 3,700 annual observations covering a wide range of political systems, institutions, exchange rate and monetary arrangements, and historic circumstances.

We also employ more recent data on external debt, including debt owed both by governments and by private entities. For emerging markets, we find that there exists a significantly more severe threshold for total gross external debt (public and private)—which is almost exclusively denominated in a foreign currency—than for total public debt (the domestically issued component of which is largely denominated in home currency). When gross external debt reaches 60 percent of GDP, annual growth declines by about two percent; for levels of external debt in excess of 90 percent of GDP, growth rates are roughly cut in half. We are not in a position to calculate separate total external debt thresholds (as opposed to public debt thresholds) for advanced countries. The available time-series is too recent, beginning only in 2000. We do note, however, that external debt levels in advanced countries now average nearly 200 percent of GDP, with external debt levels being particularly high across Europe.

The focus of this paper is on the longer term macroeconomic implications of much higher public and external debt. The final section, however, suggests an empirical investigation of the long-term effects of higher debt levels.
The Reinhart-Rogoff Factor

FAQ: Reinhart, Rogoff, and the Excel Error That Changed History

The Excel Depression

By PAUL KRUGMAN
Published: April 18, 2013 | 470 Comments

In this age of information, math errors can lead to disaster. NASA’s Mars Orbiter crashed because engineers forgot to convert to metric measurements; JPMorgan Chase’s “London Whale” venture went bad in part because modelers divided by a sum instead of an average. So, did an Excel coding error destroy the economies of the Western world?

The story so far: At the beginning of 2010, two Harvard economists, Carmen Reinhart and Kenneth Rogoff, circulated a paper, “Growth in a Time of Debt,” that purported to identify a critical “threshold,” a tipping point, for government indebtedness. Once debt exceeds 90 percent of gross domestic product, they claimed, economic growth drops off sharply.

Ms. Reinhart and Mr. Rogoff had credibility thanks to a widely admired earlier book on the history of financial crises. But now, in May 2013, a well respected economic research institute, the National Bureau of Economic Research, issued a statement saying the Reinhart-Rogoff analysis was fundamentally flawed and that the coefficient of determination was overestimated. The heroic thought experiment that had supposedly explained the financial crisis was dethroned as a result of one seemingly innocent spreadsheet error.
Excel spreadsheet error blamed for UK’s 16,000 missing coronavirus cases

The case went missing after the spreadsheet hit its filesize limit

By James Vincent  |  Oct 5, 2020, 9:41am EDT

Covid-19: Only half of 16 000 patients missed from England’s official figures have been contacted

Elisabeth Mahase

Details of nearly 16 000 cases of covid-19 were not transferred to England’s NHS Test and Trace service and were missed from official figures because of an error in the process for updating the data.

England’s health and social care secretary, Matt Hancock, told the House of Commons on Monday 5 October that after the error was discovered on Friday 2 October “6500 hours of extra contact tracing” had been carried out over the weekend. But as at Monday morning only half (51%) of the people had been reached by contact tracers.

In response, Labour’s shadow health secretary, Jonathan Ashworth, said, “Thousands of people are blissfully unaware they have been exposed to covid, spreading this deadly virus at a time when we are in the middle of winter and where even snow is much needed, and we are in the process of such an important task. This is wholly unforgivable.”

The error came as the Labour Party’s leader, Keir Starmer, said that the prime minister had “lost control” of covid-19, with no clear strategy for beating it. Speaking to the Observer, Starmer set out his five point plan for covid-19, which starts with publishing the criteria for local restrictions, as the German government did. Secondly, he said public health messaging should be improved by adding a feature to the NHS covid-19 app so people can search their postcode and find out their local restrictions.

Starmer has also said he would fix the contact tracing system by investing in NHS and university laboratories to expand testing and at the same time put local public health teams in charge of contact tracing in their areas. Routine regular testing in high transmission areas would be introduced, with results within 24 hours.
Example

```python
english = bool(input())
math = bool(input())
science = bool(input())
bonus = bool(input())

passing = True
if not english:
    english = False  # ERROR: english SHOULD BE passing
if not math:
    passing = False or bonus
if not math:
    passing = False or bonus

print(passing)  # OUTPUT VARIABLES
```

the input variables `english` and `science` are unused
Maximal Trace Semantics

Least fixpoint formulation of maximal traces

Let $\text{lfp } F_S$ be a least fixpoint formulation for whole $M_\infty$, merge finite and infinite maximal trace least fixpoint forms.

Fixpoint fusion

$M_\infty \cap \Sigma^*$ is best defined on $(\mathcal{P}(\Sigma^*), \subseteq, \cup, \cap, \emptyset, \Sigma^*)$.

$M_\infty \cap \Sigma^*$ is best defined on $(\mathcal{P}(\Sigma^*), \supseteq, \cap, \cup, \Sigma^*, \emptyset)$, the dual lattice.

(we transform the greatest fixpoint into a least fixpoint!)

We mix them into a new complete lattice $(\mathcal{P}(\Sigma^\infty), \subseteq, \cup, \cap, \emptyset, \Sigma^\infty, \emptyset)$:

- $A \subseteq B \overset{\text{def}}{\iff} (A \cap \Sigma^*) \subseteq (B \cap \Sigma^*) \land (A \cap \Sigma^\infty) \supseteq (B \cap \Sigma^\infty)$
- $A \cup B \overset{\text{def}}{=} ((A \cap \Sigma^*) \cup (B \cap \Sigma^*)) \cup ((A \cap \Sigma^\infty) \cap (B \cap \Sigma^\infty))$
- $\bot \overset{\text{def}}{=} \Sigma^\infty$
- $\top \overset{\text{def}}{=} \Sigma^*$

In this lattice, $M_\infty = \text{lfp } F_S$ where $F_S(T) \overset{\text{def}}{=} B \cup \tau \cap \tau$.

(proof on next slides)
Maximal Trace Semantics

passing = True
if not english:
    english = False
if not math:
    passing = False or bonus
if not math:
    passing = False or bonus

passing = True
english → _
math → T
science → _
bonus → _
passing → ?

passing = True
english → _
math → T
science → _
bonus → _
passing → T

passing = True
english → _
math → F
science → _
bonus → T
passing → T

passing = True
english → _
math → F
science → _
bonus → T
passing → T

passing = False or bonus
english → _
math → F
science → _
bonus → T
passing → T

passing = False or bonus
english → _
math → F
science → _
bonus → T
passing → T

passing = False or bonus
english → _
math → F
science → _
bonus → T
passing → T

passing = False or bonus
english → _
math → F
science → _
bonus → T
passing → T
Input Data (Non-)Usage

\[ \mathcal{N}_J \overset{\text{def}}{=} \{ [[P]] \in \mathcal{P}(\Sigma^{+\infty}) \mid \forall i \in J \subseteq I_P: \text{UNUSED}_i([[P]]) \} \]

\( \mathcal{N}_J \) is the set of all programs \( P \) (or, rather, their semantics \( [[P]] \)) that do not use the value of the input variables in \( J \subseteq I_P \).
Input Data (Non-)Usage

$$\mathcal{N}_J \overset{\text{def}}{=} \{ [[P]] \in \mathcal{P}(\Sigma^{+\infty}) \mid \forall i \in J \subseteq I_P : \text{UNUSED}_i([[P]]) \}$$

$$\mathcal{N}_J$$ is the set of all programs $$P$$ (or, rather, their semantics $$[[P]]$$) that **do not use** the value of the input variables in $$J \subseteq I_P$$.

$$\text{UNUSED}_i([[M]]) \overset{\text{def}}{=} \forall t \in [[P]], v \in \mathcal{V} : t_0(i) \neq v \Rightarrow \exists t' \in [[P]] :$$

$$(\forall 0 \leq j \leq |I_P| : j \neq i \Rightarrow t_0(j) = t'_0(j))$$

$$\wedge t'_0(i) = v$$

$$\wedge t'_\omega = t'_\omega$$
**Input Data (Non-)Usage**

\[ \mathcal{N}_J \overset{\text{def}}{=} \{ [[P]] \in \mathcal{P}(\Sigma^{+\infty}) \mid \forall i \in J \subseteq I_P : \text{UNUSED}_i([[[P]])} \]

\( \mathcal{N}_J \) is the set of all programs \( P \) (or, rather, their semantics \( [[P]] \)) that do not use the value of the input variables in \( J \subseteq I_P \).

\[ \text{UNUSED}_i([[[M]]) \overset{\text{def}}{=} \forall t \in [[P]], v \in \mathcal{V} : t_0(i) \neq v \Rightarrow \exists t' \in [[P]] : \\
(\forall 0 \leq j \leq |I_P| : j \neq i \Rightarrow t_0(j) = t'_0(j)) \\
\wedge t'_0(i) = v \\
\wedge t_\omega = t'_\omega \]

Intuitively: any possible program outcome is possible from any value of the input variable \( i \).
passing = True
if not english:
    english = False
if not math:
    passing = False or bonus
if not math:
    passing = False or bonus

only the input value 'math → F' yields the outcome 'passing → T'
Input Data (Non-)Usage

\[ \mathcal{N}_J \overset{\text{def}}{=} \{ [[P]] \in \mathcal{P}(\Sigma^+) \mid \forall i \in J \subseteq I_P: \text{UNUSED}_i([[P]]) \} \]

\( \mathcal{N}_J \) is the set of all programs \( P \) (or, rather, their semantics \([[P]])\) that do not use the value of the input variables in \( J \subseteq I_P \)

\[ \text{UNUSED}_i([[M]]) \overset{\text{def}}{=} \forall t \in [[P]], v \in \mathbb{V}: t_0(i) \neq v \Rightarrow \exists t' \in [[P]]: \\
(\forall 0 \leq j \leq |I_P|: j \neq i \Rightarrow t_0(j) = t'_0(j)) \\
\land t'_0(i) = v \\
\land t'_\omega = t_\omega \]

Intuitively: any possible program outcome is possible from any value of the input variable \( i \).
Trace Properties

\[[P] \subseteq [P]^{\downarrow}\]

\[[P]^{\downarrow} \subseteq \mathcal{T}\]

\[[P] \subseteq \mathcal{T}\]

**General and restricted trace properties**

**Restricted properties**

Reasoning on (and abstracting) \(\mathcal{P}(\mathcal{P}(\Sigma^*))\) is hard!

In the following, we use a simpler setting:
- a property is a set of traces \(P \in \mathcal{P}(\Sigma^*)\)
- the collecting semantics is a set of traces: \(\text{Col}(\text{prog}) \overset{\text{def}}{=} \llbracket \text{prog} \rrbracket\)
- the verification problem remains an inclusion checking: \(\llbracket \text{prog} \rrbracket \subseteq P\)
- abstraction will over-approximate the set of traces \(\llbracket \text{prog} \rrbracket\)

Example properties:
- state property \(P \overset{\text{def}}{=} S^*\) (remain in the set \(S\) of safe states)
- maximal execution time: \(P \overset{\text{def}}{=} S^{\leq k}\)
- ordering: \(P \overset{\text{def}}{=} (\Sigma \setminus \{b\})^* \cdot a \cdot \Sigma^* \cdot b \cdot \Sigma^*\) (a occurs before b)
Program Properties

General properties

General setting:
- given a program $\text{prog} \in \text{Prog}$
- its semantics: $[\cdot] : \text{Prog} \to \mathcal{P}(\Sigma^*)$ is a set of finite traces
- a property $P$ is the set of correct program semantics i.e., a set of sets of traces $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$
- gives an information order on properties
- $P \subseteq P'$ means that $P'$ is weaker than $P$ (allows more semantics)
Subset-Closed Properties

\[ [P] \subseteq [P]^{\uparrow} \]

\[ [P]^{\uparrow} \in \mathcal{H} \]

\[ \forall x \in [P]^{\uparrow} : x \in \mathcal{H} \]

\[ [P] \in \mathcal{H} \]
Input Data (Non-)Usage

\[
passing = \text{True} \\
\text{if not } \text{english:} \\
\quad \text{english} = \text{False} \\
\text{if not } \text{math:} \\
\quad \text{passing} = \text{False or bonus} \\
\text{if not } \text{math:} \\
\quad \text{passing} = \text{False or bonus}
\]

\[
\begin{align*}
\text{english} & \rightarrow _- \\
\text{math} & \rightarrow T \\
\text{science} & \rightarrow _- \\
\text{bonus} & \rightarrow \text{T or bonus} \\
\text{passing} & \rightarrow T
\end{align*}
\]

\[
\begin{align*}
\text{english} & \rightarrow _- \\
\text{math} & \rightarrow T \\
\text{science} & \rightarrow _- \\
\text{bonus} & \rightarrow \text{bonus} \\
\text{passing} & \rightarrow ?
\end{align*}
\]

\[
\begin{align*}
\text{english} & \rightarrow _- \\
\text{math} & \rightarrow F \\
\text{science} & \rightarrow _- \\
\text{bonus} & \rightarrow \text{T} \\
\text{passing} & \rightarrow ?
\end{align*}
\]

\[
\begin{align*}
\text{english} & \rightarrow _- \\
\text{math} & \rightarrow F \\
\text{science} & \rightarrow _- \\
\text{bonus} & \rightarrow \text{T} \\
\text{passing} & \rightarrow T
\end{align*}
\]

Not a subset-closed property!
Input Data (Non-)Usage

\[ \mathcal{N}_J \overset{\text{def}}{=} \{ [[P]] \in \mathcal{P}(\Sigma^{+\infty}) \mid \forall i \in J \subseteq I_P : \text{UNUSED}_i([[P]]) \} \]

\( \mathcal{N}_J \) is the set of all programs \( P \) (or, rather, their semantics \( [[P]] \)) that **do not use** the value of the input variables in \( J \subseteq I_P \)

\[ \text{UNUSED}_i([[M]]) \overset{\text{def}}{=} \forall t \in [[P]], v \in \mathcal{V} : t_0(i) \neq v \Rightarrow \exists t' \in [[P]] : \]
\[ (\forall 0 \leq j \leq |I_P| : j \neq i \Rightarrow t_0(j) = t'_0(j)) \]
\[ \land t'_0(i) = v \]
\[ \land t'_\omega = t'\omega \]

Intuitively: any possible program **outcome** is possible from any value of the input variable \( i \).

**Theorem**

\[ P \vdash \mathcal{N}_J \iff \{ [[P]] \} \subseteq \mathcal{N}_J \]
Collecting Semantics

The collecting semantics $\text{Col} : \text{Prog} \to \mathcal{P}(\mathcal{P}(\Sigma^*))$ is the strongest property of a program. Hence:

$\text{Col}(\text{prog}) \overset{\text{def}}{=} \{[[\text{prog}]]\}$

**Benefit:**

- Given a program $\text{prog}$ and a property $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$, the verification problem is an inclusion checking:
  
  $\text{Col}(\text{prog}) \subseteq P$

- Generally, the collecting semantics cannot be computed, we settle for a weaker property $S^\sharp$ that:
  - Is sound: $\text{Col}(\text{prog}) \subseteq S^\sharp$
  - Implies the desired property: $S^\sharp \subseteq P$
Collecting Semantics

Intuition

Property (by extension): set of elements that have that property

Property “being Patrick Cousot”

Property “being program P”

\{ [\lceil P \rceil] \}
(Another) **Hierarchy of Semantics**

\[
\begin{align*}
\{ [[ \mathcal{P} ]] \} & \quad \text{collecting semantics} \\
[[ \mathcal{P} ]] & \quad \text{outcome semantics} \\
\mathcal{P} & \quad \text{dependency semantics}
\end{align*}
\]
(Another) **Hierarchy of Semantics**

```
[[P]] \sim 
\alpha \sim 
[[P]]. \sim 
\alpha 
\{[[P]]\} \sim 
```

- **dependency semantics**
- **outcome semantics**
- **collecting semantics**
Outcome Semantics

partitioning a set of traces that satisfies input data (non-)usage with respect to the program outcome yields sets of traces that also satisfy input data (non-)usage.
Outcome Semantics

\[ \bigcirc \overset{\text{def}}{=} \{ \sum_{o_1 = v_1, \ldots, o_k = v_k}^+ \mid v_1, \ldots, v_k \in \mathcal{V} \} \cup \{ \Sigma^\omega \} \]

Lemma

\[ P \models \mathcal{N}_j \iff \{ [[P]] \cap O \mid O \in \bigcirc \} \subseteq \mathcal{N}_j \]

input data (non-) usage can be decided independently for each possible outcome.
**Outcome Semantics**

\[ \emptyset \overset{\text{def}}{=} \{ \Sigma_{o_1=v_1, \ldots, o_k=v_k} \mid v_1, \ldots, v_k \in \mathcal{V} \} \cup \{ \Sigma^\omega \} \]

**Lemma**

\[ P \models \mathcal{N}_J \iff \{ [[P]] \cap O \mid O \in \emptyset \} \subseteq \mathcal{N}_J \]

\[ \langle \mathcal{P}(\mathcal{P}(\Sigma^{+\infty})), \subseteq \rangle \quad \langle \mathcal{P}(\mathcal{P}(\Sigma^{+\infty})), \subseteq. \rangle \]

\[ \alpha_*(S) \overset{\text{def}}{=} \{ T \cap O \mid T \in S \land O \in \emptyset \} \]
passing = True
if not english:
    english = False
if not math:
    passing = False or bonus
if not math:
    passing = False or bonus
**Outcome Semantics**

\[ \mathcal{O} \overset{\text{def}}{=} \{ \Sigma_{o_1=v_1, \ldots, o_k=v_k} \mid v_1, \ldots, v_k \in \mathcal{V} \} \cup \{ \Sigma^\omega \} \]

**Lemma**

\[ P \models \mathcal{N}_j \iff \{ \llbracket P \rrbracket \cap O \mid O \in \mathcal{O} \} \subseteq \mathcal{N}_j \]

\[ \langle \mathcal{P}(\mathcal{P}(\Sigma^{+\infty})), \subseteq \rangle \quad \langle \mathcal{P}(\mathcal{P}(\Sigma^{+\infty})), \subseteq. \rangle \]

**Outcome Abstraction**

\[ \alpha.(S) \overset{\text{def}}{=} \{ T \cap O \mid T \in S \land O \in \mathcal{O} \} \]

\[ \llbracket P \rrbracket. \overset{\text{def}}{=} \alpha.(\{ \llbracket P \rrbracket \}) = \{ \llbracket P \rrbracket \cap O \mid O \in \mathcal{O} \} \]
Maximal Trace Semantics

Least fixpoint formulation of maximal traces

Idea:
To get a least fixpoint formulation for whole \( \mathcal{M} \), merge finite and infinite maximal trace least fixpoint forms.

Fixpoint fusion
\( \mathcal{M} = \text{fl} \mathcal{U} \) is best defined on \((\mathcal{P}(\mathcal{U}), \subseteq, \mathcal{F}, \text{fl}, \bigcap, \mathcal{U})\), the dual lattice (we transform the greatest fixpoint into a least fixpoint!)

We mix them into a new complete lattice \((\mathcal{P}(\mathcal{U} \cup \mathcal{E}), \subseteq, \mathcal{F}, \text{fl}, \bigcap, \mathcal{U})\):

\[
A \subseteq B \iff (A \cap \mathcal{U}) \subseteq (B \cap \mathcal{U})
\]
\[
A \triangleq B \iff ((A \cap \mathcal{U}) \subseteq (B \cap \mathcal{U})) \land ((A \cap \mathcal{E}) \subseteq (B \cap \mathcal{E}))
\]
\[
A \cup B \iff ((A \cap \mathcal{U}) \cup (B \cap \mathcal{U}) \subseteq (A \cup B \cap \mathcal{U}))
\]

\[
A \subseteq B \implies A \cup B = B
\]

\[
A \cap B = A \cap B \cap \mathcal{U}
\]

\[
\mathcal{U} = \mathcal{U} \cap \mathcal{E}
\]

\[
\mathcal{E} = \mathcal{E} \cap \mathcal{U}
\]

\[
\bigcap \mathcal{U} = \bigcap \mathcal{U} \cap \mathcal{E}
\]

\[
\bigcup \mathcal{U} = \bigcup \mathcal{U} \cap \mathcal{E}
\]

\[
\mathcal{U} \cup \mathcal{E} = \mathcal{U} \cup \mathcal{E} \cap \mathcal{U}
\]

\[
\mathcal{E} \cap \mathcal{U} = \mathcal{E} \cap \mathcal{U} \cap \mathcal{E}
\]

\[
\mathcal{U} \cap \mathcal{E} = \mathcal{U} \cap \mathcal{E} \cap \mathcal{U}
\]

\[
\text{lfp}(\mathcal{M}) = \text{fl} \mathcal{U} \text{fp}(\mathcal{M})
\]

Proof on next slides

\[
S_0 = \{\mathcal{U}^w\}, \emptyset
\]

\[
S_1 = \{\mathcal{V}_{\mathcal{U}^w} \mid v \in \mathcal{V}\} \cup \{\mathcal{T} \mathcal{U}^w\}
\]

\[
S_2 = \{\mathcal{V}_{\mathcal{U}^w} \cup \{\mathcal{T} \mathcal{V}_{\mathcal{U}^w}\} \mid v \in \mathcal{V}\} \cup \{\mathcal{T} \mathcal{V}_{\mathcal{U}^w}\}
\]

\[
\mathcal{U}^w = \text{fl} \mathcal{U}^w
\]

\[
\mathcal{T} \mathcal{U}^w = \text{fl} \mathcal{T} \mathcal{U}^w
\]
### Outcome Semantics

\[
S_1 \sqsubseteq S_2 \overset{\text{def}}{=} \bigwedge_{v_1, \ldots, v_k \in V} S_{1, o_1=v_1, \ldots, o_k=v_k} \sqsubseteq S_{2, o_1=v_1, \ldots, o_k=v_k} \wedge S_1^\omega \supseteq S_2^\omega
\]

**Theorem 1.** The outcome semantics \( \Lambda_\bullet \in \mathcal{P}(\mathcal{P}(\Sigma^{+\infty})) \) can be expressed as a least fixpoint in \( \langle \mathcal{P}(\mathcal{P}(\Sigma^{+\infty})), \sqsubseteq, \sqcup, \sqcap, \{\Sigma^\omega, \emptyset\}, \{\emptyset, \Sigma^+\rangle \) as:

\[
\Lambda_\bullet = \text{lfp} \sqsubseteq \Theta_\bullet
\]

\[
\Theta_\bullet(S) \overset{\text{def}}{=} \{\Omega_{o_1=v_1, \ldots, o_k=v_k} \mid v_1, \ldots, v_k \in V\} \cup \{\tau ; T \mid T \in S\}
\]

where \( S_1 \cup S_2 \overset{\text{def}}{=} \{S_{1, o_1=v_1, \ldots, o_k=v_k} \cup S_{2, o_1=v_1, \ldots, o_k=v_k} \mid v_1, \ldots, v_k \in V\} \cup S_1^\omega \cup S_2^\omega \).

(proof by Kleenian Fixpoint Transfer [Urban18])
Input Data (Non-)Usage

\[ \mathcal{N}_J \overset{\text{def}}{=} \{ [[P]] \in \mathcal{P}(\Sigma^{+\infty}) \mid \forall i \in J \subseteq I_P : \text{UNUSED}_i([[P]]) \} \]

\( \mathcal{N}_J \) is the set of all programs \( P \) (or, rather, their semantics \( [[P]] \)) that do not use the value of the input variables in \( J \subseteq I_P \).

**Outcome Semantics**

\( O \overset{\text{def}}{=} \{ \Sigma^{+}_{v_1,\ldots,v_k} \mid v_1,\ldots,v_k \in \mathcal{V}^* \} \cup \{ \Sigma^o \} \)

**Lemma**

\( P \models \mathcal{N}_J \iff \{ [[P]] \cap O \mid O \in O \} \subseteq \mathcal{N}_J \)

\( \alpha(S) \overset{\text{def}}{=} \{ T \cap O \mid T \in S \land O \in O \} \)

\( [[P]] \overset{\text{def}}{=} \alpha_*(\{ [[P]] \}) = \{ [[P]] \cap O \mid O \in O \} \)

**Theorem**

\( P \models \mathcal{N}_J \iff \{ [[P]] \} \subseteq \mathcal{N}_J \iff \alpha_*(\{ [[P]] \}) \subseteq \mathcal{N}_J \iff \mathcal{N}_J \iff [[P]] \subseteq \mathcal{N}_J \)
(Another) Hierarchy of Semantics

dependency semantics

outcome semantics

collecting semantics
Dependency Semantics

...to reason about input data (non-)usage we do not need to consider all intermediate computations between the initial and final states of a trace (if any)
Dependency Semantics

\[ \langle \mathcal{P}(\mathcal{P}(\Sigma^{+\infty})), \sqsubseteq \rangle \quad \langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma_\perp)), \sqsubseteq \rangle \]

\[ \gamma_\sim \quad \alpha_\sim \]

\[ \alpha_\sim(S) \overset{\text{def}}{=} \{ \langle t_0, t_\omega \rangle \in \Sigma \times \Sigma_\perp \mid t \in T \mid T \in S \} \]

\[ \chi_\sim(S) \overset{\text{def}}{=} \{ T \in \mathcal{P}(\Sigma^{+\infty}) \mid \{ \langle t_0, t_\omega \rangle \in \Sigma \times \Sigma_\perp \mid t \in T \} \in S \} \]

\[ \Sigma_\perp \overset{\text{def}}{=} \Sigma \cup \{ \perp \} \]
Dependency Semantics

passing = True
if not english:
    english = False
if not math:
    passing = False or bonus
if not math:
    passing = False or bonus

english → _
math → T
science → _
bonus → _
passing → ?
Dependency Semantics

\[
\begin{align*}
\alpha_{\leadsto}(S) & \overset{\text{def}}{=} \{ \{ \langle t_0, t_\omega \rangle \in \Sigma \times \Sigma_\bot \mid t \in T \} \mid T \in S \} \\
\gamma_{\leadsto}(S) & \overset{\text{def}}{=} \{ T \in \mathcal{P}(\Sigma^{+\infty}) \mid \{ \langle t_0, t_\omega \rangle \in \Sigma \times \Sigma_\bot \mid t \in T \} \in S \} \\
[[P]]_{\leadsto} & \overset{\text{def}}{=} \alpha_{\leadsto}([[P]])_\cdot = \{ \{ t_0, t_\omega \} \in \Sigma \times \Sigma \mid t \in [[P]] \cap O \} \mid O \in \mathcal{O} \}
\end{align*}
\]
the outcome semantics $A_\bullet$ can be equivalently expressed as follows:

$$A_\bullet = A_\Sigma^+ \cup A_\Sigma^\omega = \text{lfp}_{\emptyset} \Theta_\Sigma^+ \cup \text{lfp}_{\{\Sigma^\omega\}} \Theta_\Sigma^\omega$$

$$\Theta_\Sigma^+(S) \overset{\text{def}}{=} \{\Omega_{o_1=v_1, \ldots, o_k=v_k} \mid v_1, \ldots, v_k \in V \} \cup \{\tau ; T \mid T \in S\} \quad (13)$$

$$\Theta_\Sigma^\omega(S) \overset{\text{def}}{=} \{\tau ; T \mid T \in S\}$$
Dependency Semantics

the outcome semantics $A_*$ can be equivalently expressed as follows:

$$A_* = A^+_* \cup A^\omega_* = \text{lfp}_\emptyset^\Sigma \Theta^+ \cup \text{lfp}_{\{\Sigma^\omega\}}^\Sigma \Theta^\omega$$

$$\Theta^+ (S) \overset{\text{def}}{=} \{\Omega_{o_1=v_1, \ldots, o_k=v_k} \mid v_1, \ldots, v_k \in V\} \cup \{\tau ; T \mid T \in S\} \tag{13}$$

$$\Theta^\omega (S) \overset{\text{def}}{=} \{\tau ; T \mid T \in S\}$$

**Lemma 2.** The abstraction $\Lambda^+_{\sim} \overset{\text{def}}{=} \alpha_{\sim} (A^+_*) \in \mathcal{P} (\mathcal{P} (\Sigma \times \Sigma))$ can be expressed as a least fixpoint in $\langle \mathcal{P} (\mathcal{P} (\Sigma \times \Sigma)), \sqsubseteq, \sqcup, \sqcap, \{\Sigma \times \{\bot\}, \emptyset\}, \{\emptyset, \Sigma \times \Sigma\} \rangle$ as:

$$A^+_{\sim} = \text{lfp}_{\emptyset}^\Sigma \Theta^+_{\sim}$$

$$\Theta^+_{\sim} (S) \overset{\text{def}}{=} \{\Omega_{o_1=v_1, \ldots, o_k=v_k} \times \Omega_{o_1=v_1, \ldots, o_k=v_k} \mid v_1, \ldots, v_k \in V\} \cup \{\tau \circ R \mid R \in S\} \tag{14}$$

**Proof (Sketch).** By Kleenian fixpoint transfer (cf. Theorem 17 in [12]).
the outcome semantics \( \Lambda_* \) can be equivalently expressed as follows:

\[
\Lambda_* = \Lambda_*^+ \cup \Lambda_*^\omega = \text{lfp}_\emptyset \Theta_*^+ \cup \text{lfp}_{\{\Sigma^\omega\}} \Theta_*^\omega
\]

\[
\Theta_*^+(S) \stackrel{\text{def}}{=} \{ \Omega_{o_1=\ldots=o_k=v_k} \mid v_1, \ldots, v_k \in V \} \cup \{ \tau ; T \mid T \in S \}
\]

\[
\Theta_*^\omega(S) \stackrel{\text{def}}{=} \{ \tau ; T \mid T \in S \}
\]

**Lemma 2.** The abstraction \( \Lambda^+_{\sim} \stackrel{\text{def}}{=} \alpha_{\sim}(\Lambda^+_*) \in \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma)) \) can be expressed as a least fixpoint in \( \langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma_\bot)) \rangle \) as:

\[
\Lambda^+_{\sim} = \text{lfp}_{\{\emptyset\}} \Theta^+_{\sim}
\]

\[
\Theta^+_{\sim}(S) \stackrel{\text{def}}{=} \{ \Omega_{o_1=\ldots=o_k=v_k} \times \Omega_{o_1=\ldots=o_k=v_k} \mid v_1, \ldots, v_k \in V \} \cup \{ \tau \circ R \mid R \in S \}
\]

**Proof (Sketch).** By Kleenian fixpoint transfer (cf. Theorem 18 in [12]).

**Lemma 3.** The abstraction \( \Lambda^\omega_{\sim} \stackrel{\text{def}}{=} \alpha_{\sim}(\Lambda^\omega_*) \in \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma)) \) can be expressed as a least fixpoint in \( \langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma_\bot)) \rangle \) as:

\[
\Lambda^\omega_{\sim} = \text{lfp}_{\{\Sigma \times \{\bot\}\}} \Theta^\omega_{\sim}
\]

\[
\Theta^\omega_{\sim}(S) \stackrel{\text{def}}{=} \{ \tau \circ R \mid R \in S \}
\]

**Proof (Sketch).** By Tarskian fixpoint transfer (cf. Theorem 18 in [12]).
Dependency Semantics

Lemma 2. The abstraction $\Lambda^+_\sim \overset{\text{def}}{=} \alpha_\sim(\Lambda^+_\bullet) \in \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma))$ can be expressed as a least fixpoint in $\langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma)) \rangle, \sqsubseteq, \sqcup, \sqcap, \{\Sigma \times \{\bot\}, \emptyset\}, \{\emptyset, \Sigma \times \Sigma\}$ as:

$$\Lambda^+_\sim = \text{lfp}_{\{\emptyset\}}^{\subseteq} \Theta^+_\sim$$

$$\Theta^+_\sim(S) \overset{\text{def}}{=} \{ \Omega_{o_1=v_1, \ldots, o_k=v_k} \times \Omega_{o_1=v_1, \ldots, o_k=v_k} \mid v_1, \ldots, v_k \in V \} \cup \{ \tau \circ R \mid R \in S \}$$

Proof (Sketch). By Tarski's fixed-point theorem.

Lemma 3. The abstraction $\Lambda^\omega_\sim \overset{\text{def}}{=} \alpha_\sim(\Lambda^\omega_\bullet) \in \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma))$ can be expressed as a least fixpoint in $\langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma)) \rangle, \sqsubseteq, \sqcup, \sqcap, \{\Sigma \times \{\bot\}, \emptyset\}, \{\emptyset, \Sigma \times \Sigma\}$ as:

$$\Lambda^\omega_\sim = \text{lfp}_{\{\Sigma \times \{\bot\}\}}^{\subseteq} \Theta^\omega_\sim$$

$$\Theta^\omega_\sim(S) \overset{\text{def}}{=} \{ \tau \circ R \mid R \in S \}$$

(15)

Proof (Sketch). By Tarski's fixed-point theorem (cf. Theorem 18 in [12]).

Theorem 3. The dependency semantics $\Lambda_\sim \in \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma))$ can be expressed as a least fixpoint in $\langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma)) \rangle, \sqsubseteq, \sqcup, \sqcap, \{\Sigma \times \{\bot\}, \emptyset\}, \{\emptyset, \Sigma \times \Sigma\}$ as:

$$\Lambda_\sim = \Lambda^+_\sim \cup \Lambda^\omega_\sim = \text{lfp}_{\{\Sigma \times \{\bot\}, \emptyset\}}^{\subseteq} \Theta_\sim$$

$$\Theta_\sim(S) \overset{\text{def}}{=} \{ \Omega_{o_1=v_1, \ldots, o_k=v_k} \times \Omega_{o_1=v_1, \ldots, o_k=v_k} \mid v_1, \ldots, v_k \in V \} \cup \{ \tau \circ R \mid R \in S \}$$

(16)

Proof (Sketch). The proof follows immediately from Lemma 2 and Lemma 3.
Input Data (Non-)Usage

\[ \mathcal{N}_J \overset{\text{def}}{=} \{ [[P]] \in \mathcal{P}(\Sigma^+) \mid \forall i \in J \subseteq I_P: \text{UNUSED}_i([[P]]) \} \]

\( \mathcal{N}_J \) is the set of all programs \( P \) (or, rather, their semantics \( [[P]] \)) that do not use the value of the input variables in \( J \subseteq I_P \)

Dependency Semantics

\[ \alpha_\rightarrow(S) \overset{\text{def}}{=} \{ \langle t_0, t_w \rangle \in \Sigma \times \Sigma \mid t \in T \mid T \in S \} \]

\[ \gamma_\rightarrow(S) \overset{\text{def}}{=} \{ T \in \mathcal{P}(\Sigma^+) \mid \{ \langle t_0, t_w \rangle \in \Sigma \times \Sigma \mid t \in T \} \in S \} \]

\[ [[P]]_\rightarrow \overset{\text{def}}{=} \alpha_\rightarrow([[P]]) = \{ \langle t_0, t_w \rangle \in \Sigma \times \Sigma \mid t \in [[P]] \cap \Omega \mid \Omega \in \Omega \} \]

Theorem

\[ P \models \mathcal{N}_J \iff \{ [[P]] \} \subseteq \mathcal{N}_J \iff [[P]]. \subseteq \mathcal{N}_J \iff \gamma_\rightarrow([[P]]_\rightarrow) \subseteq \mathcal{N}_J \]

Outcome Semantics

\[ t_0(i) \neq v \Rightarrow \exists t' \in [[P]]: i \Rightarrow t_0(j) = t'_0(j) \]
Input Data (Non-)Usage Abstractions

Over-Approximation of the Used Input Data

⇒ Under-Approximation of the Unused Input Data

\[ P \models \mathcal{N}_{\bar{J} \subseteq J} \iff \gamma \sim_{\alpha} (\gamma_A(\llbracket P \rrbracket_A)) \subseteq \mathcal{N}_{\bar{J} \subseteq J} \]

- soundness
- collecting semantics \( \alpha \)
- outcome semantics \( \alpha \)
- dependency semantics
- secure information flow
- strong-liveness
- syntactic data (non-)usage
- categorization
- liveness
- syntactic data
- non-usage
Secure Information Flow

possibilistic non-interference coincides with input data (non-)usage when the set $J$ of unused input variables contains all input variables:
- input variables are high-security variables
- output variables are low-security variables

\[ S \xrightarrow{L \rightarrow} x \]
\[ S \xrightarrow{L \rightarrow} y \]
\[ H \rightarrow t \]
\[ S \xrightarrow{L \rightarrow} z \]
\[ H \rightarrow w \]

\[ [P]_F \]

Abstract

A key feature of static analysis for secure information flow can be expressed and proved correct entirely within the framework of abstract interpretation. The key idea is to define a Galois connection that classifies the simplest form of noninterference. From such Galois connections, we introduce a fixpoint characterisation of such Galois connections, we introduce a fixpoint characterisation of all properties entirely within the calculational framework of abstract interpretation. We evaluate this technique by deriving example static analysis similar to the logic of Amtoft and Banerjee (SAS'04) and information flow, we derive a novel cardinality analysis that bounds the k-safety. We put the framework to use and introduce variations that achieve precision rivalling the most recent and precise static analyses for information flow.

\[ e ::= v \mid x \mid \text{not } e \mid e \text{ and } e \mid e \text{ or } e \]
\[ s ::= \text{skip} \mid x = e \mid \text{if } c : s \text{ else: } s \mid \text{while } c : s \mid s s \] (expressions)

\[ \Theta_F[\text{skip}](S) \overset{\text{def}}{=} S \]
\[ \Theta_F[x = e](S) \overset{\text{def}}{=} \{ L \rightarrow y \in S \mid y \neq x \} \cup \{ L \rightarrow x \mid \forall_F[e]S \} \]
\[ \Theta_F[\text{if } c : s_1 \text{ else: } s_2](S) \overset{\text{def}}{=} \begin{cases} \Theta_F[s_1](S) \cup_F \Theta_F[s_2](S) & \text{if } \forall_F[e]S \\ \{ L \rightarrow x \in S \mid x \notin w(s_1) \cup w(s_2) \} & \text{otherwise} \end{cases} \]
\[ \Theta_F[\text{while } e : s](S) \overset{\text{def}}{=} \text{lfp}_{\leq_F} \Theta_F[\text{if } c : s \text{ else:skip}] \]
\[ \Theta_F[s_1 s_2](S) \overset{\text{def}}{=} \Theta_F[s_2] \circ \Theta_F[s_1](S) \]

\[ S \text{ guarantees a unique value for each independently of values of input variables} \]
\[ \forall_F[\text{x}]: S \iff L \rightarrow x \in S \]

set of variables modified by $s_i$

set of security levels

abstract domain

dependencies subject to more than

program is correct if all its traces satisfy the predicate. By c
with such trace properties, extensional definitions of depend-
i ve more than one trace. To express that the final va-
variable $x$ may depend only on the initial value of a variable $y$
requirement—known as noninterference in the security
(Sabelfeld and Myers 2003)—is that any two traces with
z
z
z
z
z
z
z
z
Secure Information Flow

possibilistic non-interference coincides with input data (non-)usage when the set $J$ of unused input variables contains all input variables:

- input variables are high-security variables
- output variables are low-security variables

$$\begin{align*}
\Theta_F[\text{skip}](S) & \overset{\text{def}}{=} S \\
\Theta_F[x = e](S) & \overset{\text{def}}{=} \{L \leadsto y \in S \mid y \neq x\} \cup \{L \leadsto x \mid V_F[e]S\} \\
\Theta_F[\text{if } c : s_1 \text{ else: } s_2](S) & \overset{\text{def}}{=} \begin{cases} 
\Theta_F[s_1](S) \cup V_F\Theta_F[s_2](S) & \text{if } V_F[e]S \\
\{L \leadsto x \in S \mid x \notin w(s_1) \cup w(s_2)\} & \text{otherwise}
\end{cases} \\
\Theta_F[\text{while } c : s](S) & \overset{\text{def}}{=} \bigcup_{S' \subseteq S} \Theta_F[\text{if } c : s \text{ else: skip}]
\end{align*}$$

$$\Theta_F[s_1 s_2](S) \overset{\text{def}}{=} \Theta_F[s_2] \circ \Theta_F[s_1](S)$$

passing = True
if not english:
   english = False
if not math:
   passing = False or bonus
if not math:
   passing = False or bonus

$$[P]_F$$
Secure Information Flow

possibilistic non-interference coincides with input data (non-)usage when the set $J$ of unused input variables contains all input variables:

- input variables are high-security variables
- output variables are low-security variables

$$e ::= \nu \mid x \mid \text{not } e \mid e \text{ and } e \mid e \text{ or } e$$

$$s ::= \text{skip} \mid x = e \mid \text{if } c : s \text{ else: } s \mid \text{while } c : s \mid s \ s$$

$$\Theta_F[\text{skip}](S) \overset{\text{def}}{=} S$$

$$\Theta_F[x = e](S) \overset{\text{def}}{=} \{ L \leadsto y \in S \mid y \neq x \} \cup \{ L \leadsto x \mid \forall F[e]S \}$$

$$\Theta_F[\text{if } c : s_1 \text{ else: } s_2](S) \overset{\text{def}}{=} \begin{cases} \Theta_F[s_1](S) \sqcup \Theta_F[s_2](S) & \text{if } \forall F[e]S \\ \{ L \leadsto x \in S \mid x \notin w(s_1) \cup w(s_2) \} & \text{otherwise} \end{cases}$$

$$\Theta_F[\text{while } e : s](S) \overset{\text{def}}{=} \text{lfp}_{S}^{\subseteq} \Theta_F[\text{if } e : s \text{ else: skip}]$$

$$\Theta_F[s_1 \ s_2](S) \overset{\text{def}}{=} \Theta_F[s_2] \circ \Theta_F[s_1](S)$$

passing = True
if not english:
    english = False
if not math:
    passing = False or bonus
if not math:
    passing = False or bonus

$\Rightarrow$ $L \rightarrow \text{passing}, (H \rightarrow \text{english, math, science, bonus})$
Secure Information Flow

possibilistic non-interference coincides with input data (non-)usage when the set J of unused input variables contains all input variables:

- input variables are high-security variables
- output variables are low-security variables

\[ e := v \mid x \mid \text{not } e \mid e \text{ and } e \mid e \text{ or } e \]
\[ s := \text{skip } \mid x = e \mid \text{if } c : s \text{ else: } s \mid \text{while } c : s \mid s \]

\[
\begin{align*}
\Theta_F[\text{skip}](S) &\overset{\text{def}}{=} S \\
\Theta_F[x = e](S) &\overset{\text{def}}{=} \{L \leadsto y \in S \mid y \neq x\} \cup \{L \leadsto x \mid \forall F[e]\} \\
\Theta_F[\text{if } c: s_1 \text{ else: } s_2](S) &\overset{\text{def}}{=} \\
&\begin{cases} \\
\Theta_F[s_1](S) \cup_F \Theta_F[s_2](S) & \text{if } \forall F[e] \\
\{L \leadsto x \in S \mid x \notin w(s_1) \cup w(s_2)\} & \text{otherwise} \\
\end{cases} \\
\Theta_F[\text{while } c : s](S) &\overset{\text{def}}{=} \text{lfp}_{S_F} \Theta_F[\text{if } c : s \text{ else: skip}] \\
\Theta_F[s_1 s_2](S) &\overset{\text{def}}{=} \Theta_F[s_2] \circ \Theta_F[s_1](S)
\end{align*}
\]

\[
\begin{align*}
\text{passing} &= \text{True} \\
\text{if not} &\text{ english: } \\
&\text{english} = \text{False} \\
\text{if not} &\text{ math: } \\
&\text{passing} = \text{False or bonus} \\
\text{if not} &\text{ math: } \\
&\text{passing} = \text{False or bonus}
\end{align*}
\]
Secure Information Flow

possibilistic non-interference coincides with input data (non-)usage when the set $J$ of unused input variables contains all input variables:

- input variables are high-security variables
- output variables are low-security variables

\[
\begin{align*}
\Theta_F[\text{skip}](S) & \overset{\text{def}}{=} S \\
\Theta_F[x \leftarrow e](S) & \overset{\text{def}}{=} \{ L \leadsto y \in S \mid y \neq x \} \cup \{ L \leadsto x \mid \forall F[e]S \} \\
\Theta_F[\text{if } c : s_1 \text{ else: } s_2](S) & \overset{\text{def}}{=} \begin{cases} 
\Theta_F[s_1](S) \cup F \Theta_F[s_2](S) & \text{if } \forall F[e]S \\
\{ L \leadsto x \in S \mid x \notin w(s_1) \cup w(s_2) \} & \text{otherwise}
\end{cases} \\
\Theta_F[\text{while } e : s](S) & \overset{\text{def}}{=} \text{lp}_{S}^{F} \Theta_F[\text{if } e : s \text{ else: skip}] \\
\Theta_F[s_1 \cdot s_2](S) & \overset{\text{def}}{=} \Theta_F[s_2] \circ \Theta_F[s_1](S)
\end{align*}
\]

passing = $\text{True}$
if not english:
  english = $\text{False}$
if not math:
  passing = $\text{False or bonus}$
if not math:
  passing = $\text{False or bonus}$
Secure Information Flow

possibilistic non-interference coincides with input data (non-)usage when the set J of unused input variables contains all input variables:

- input variables are high-security variables
- output variables are low-security variables

$$e ::= \nu \mid x \mid \text{not } e \mid e \text{ and } e \mid e \text{ or } e$$

$$s ::= \text{skip} \mid x \leftarrow e \mid \text{if } c: s \text{ else: } s \mid \text{while } c: s \mid s \ s$$

$$\Theta_F[\text{skip}](S) \overset{\text{def}}{=} S$$
$$\Theta_F[x \leftarrow e](S) \overset{\text{def}}{=} \{L \rightsquigarrow y \in S \mid y \neq x\} \cup \{L \rightsquigarrow x \mid \Theta_F[e][S]\}$$
$$\Theta_F[\text{if } c: s_1 \text{ else: } s_2](S) \overset{\text{def}}{=} \begin{cases} \Theta_F[s_1](S) \cup \Theta_F[s_2](S) & \text{if } \Theta_F[e][S] \\ \{L \rightsquigarrow x \in S \mid x \not\in w(s_1) \cup w(s_2)\} & \text{otherwise} \end{cases}$$
$$\Theta_F[\text{while } c: s](S) \overset{\text{def}}{=} \text{lfp}^{\Theta_F} \Theta_F[\text{if } c: s \text{ else: skip}]$$
$$\Theta_F[s_1 s_2](S) \overset{\text{def}}{=} \Theta_F[s_2] \circ \Theta_F[s_1](S)$$

Passing = True
if not english:
    english = False
if not math:
    passing = False or bonus
if not math:
    passing = False or bonus

L → passing, (H → english, math, science, bonus)
L → passing, (H → english, math, science, bonus)
L → passing, (H → english, math, science, bonus)
(H → english, math, science, bonus, passing)
Secure Information Flow

possibilistic non-interference coincides with input data (non-)usage when the set $J$ of unused input variables contains all input variables:

- input variables are high-security variables
- output variables are low-security variables

\[
\begin{align*}
\Theta_F[\text{skip}](S) & \overset{\text{def}}{=} S \\
\Theta_F[x = e](S) & \overset{\text{def}}{=} \{ L \leadsto y \in S \mid y \neq x \} \cup \{ L \leadsto x \mid \forall F[e]S \} \\
\Theta_F[\text{if } c : s_1 \text{ else: } s_2](S) & \overset{\text{def}}{=} \begin{cases} 
\Theta_F[s_1](S) \cup_F \Theta_F[s_2](S) & \text{if } \forall F[e]S \\
\{ L \leadsto x \in S \mid x \not\in w(s_1) \cup w(s_2) \} & \text{otherwise}
\end{cases} \\
\Theta_F[\text{while } e : s](S) & \overset{\text{def}}{=} \text{lfp}_S \Theta_F[\text{if } e : s \text{ else: skip}] \\
\Theta_F[s_1, s_2](S) & \overset{\text{def}}{=} \Theta_F[s_2] \circ \Theta_F[s_1](S)
\end{align*}
\]

\[
\begin{align*}
\text{passing} &= \text{True} \\
\text{if not english:} & \quad \text{english} = \text{False} \\
\text{if not math:} & \quad \text{passing} = \text{False or bonus} \\
\text{if not math:} & \quad \text{passing} = \text{False or bonus}
\end{align*}
\]

\[
\begin{align*}
L \rightarrow x & \quad (H \rightarrow \text{english, math, science, bonus}) \\
L \rightarrow y & \quad (H \rightarrow \text{english, math, science, bonus}) \\
L \rightarrow z & \quad (H \rightarrow \text{english, math, science, bonus}) \\
H \rightarrow t & \quad (H \rightarrow \text{english, math, science, bonus, passing}) \\
H \rightarrow w & \quad (H \rightarrow \text{english, math, science, bonus, passing})
\end{align*}
\]
Secure Information Flow

possibilistic non-interference coincides with input data (non-)usage when the set J of unused input variables contains all input variables:

- input variables are high-security variables
- output variables are low-security variables

and the program is terminating.

\[ e := v | x | not e | e and e | e or e \]
\[ s := skip | x = e | if e: s else: s | while e: s | s s \]

Theorem

\[ P \models \mathcal{N}_I^* \iff \gamma_\sim(\gamma_F([P]_F)) \subseteq \mathcal{N}_I^* \]
a variable is strongly live if
- it is used in an assignment to another strongly live variable
- it is used in a statement other than an assignment

\[
\begin{align*}
\Theta_X[\text{skip}](S) & \overset{\text{def}}{=} S \\
\Theta_X[x = e](S) & \overset{\text{def}}{=} \begin{cases} 
(S \setminus \{x\}) \cup \text{VARS}(e) & x \in S \\
S & \text{otherwise}
\end{cases} \\
\Theta_X[\text{if } b: s_1 \text{ else } s_2](S) & \overset{\text{def}}{=} \text{VARS}(b) \cup \Theta_X[s_1](S) \cup \Theta_X[s_2](S) \\
\Theta_X[\text{while } b: s](S) & \overset{\text{def}}{=} \text{VARS}(b) \cup \Theta_X[s](S) \\
\Theta_X[s_1 s_2](S) & \overset{\text{def}}{=} \Theta_X[s_1] \circ \Theta_X[s_2](S)
\end{align*}
\]

\[\langle \mathcal{P}(X), \mathcal{L}, \mathcal{U}, \sqcup, \emptyset, x \rangle : \text{abstract domain}\]
Strong-Liveness

A variable is strongly live if:
- it is used in an assignment to another strongly live variable
- it is used in a statement other than an assignment

\[
\begin{align*}
\Theta_X[\text{skip}](S) & \overset{\text{def}}{=} S \\
\Theta_X[x = e](S) & \overset{\text{def}}{=} \begin{cases} 
(S \setminus \{x\}) \cup \text{VARS}(e) & x \in S \\
S & \text{otherwise}
\end{cases} \\
\Theta_X[\text{if } b: s_1 \text{ else: } s_2](S) & \overset{\text{def}}{=} \text{VARS}(b) \cup \Theta_X[s_1](S) \cup \Theta_X[s_2](S) \\
\Theta_X[\text{while } b: s](S) & \overset{\text{def}}{=} \text{VARS}(b) \cup \Theta_X[s](S) \\
\Theta_X[s_1 s_2](S) & \overset{\text{def}}{=} \Theta_X[s_1] \circ \Theta_X[s_2](S)
\end{align*}
\]

\[
e ::= v \mid x \mid \text{not } e \mid e \text{ and } e \mid e \text{ or } e \\
s ::= \text{skip} \mid x = e \mid \text{if } c: s \text{ else: } s \mid \text{while } c: s \mid s s
\]

(Expressions)

(Statements)

\$
\begin{align*}
passing & = \text{True} \\
\text{if not } \text{english:} & \\
& \text{english } = \text{False} \\
\text{if not } \text{math:} & \\
& \text{passing } = \text{False or bonus} \\
\text{if not } \text{math:} & \\
& \text{passing } = \text{False or bonus}
\end{align*}
\$

\{ passing \}

The initial set of strongly live variables is the set of output variables.
a variable is **strongly live** if
- it is used in an assignment to another strongly live variable
- it is used in a statement other than an assignment

\[
\begin{align*}
\Theta_X[\text{skip}](S) & \defeq S \\
\Theta_X[x = e](S) & \defeq \begin{cases} 
(S \setminus \{x\}) \cup \text{VARS}(e) & x \in S \\
S & \text{otherwise}
\end{cases} \\
\Theta_X[\text{if } b : s_1 \text{ else: } s_2](S) & \defeq \text{VARS}(b) \cup \Theta_X[s_1](S) \cup \Theta_X[s_2](S) \\
\Theta_X[\text{while } b : s](S) & \defeq \text{VARS}(b) \cup \Theta_X[s](S) \\
\Theta_X[s_1 s_2](S) & \defeq \Theta_X[s_1] \circ \Theta_X[s_2](S)
\end{align*}
\]

passing = `True`
if not english:
  english = `False`
if not math:
  passing = `False` or bonus
if not math:
  passing = `False` or bonus
Strong-Liveness

a variable is strongly live if
- it is used in an assignment to another strongly live variable
- it is used in a statement other than an assignment

\[
\begin{align*}
\Theta_X[\text{skip}](S) & \overset{\text{def}}{=} S \\
\Theta_X[x := e](S) & \overset{\text{def}}{=} \begin{cases} 
(S \setminus \{x\}) \cup \text{VARS}(e) & x \in S \\
S & \text{otherwise}
\end{cases} \\
\Theta_X[\text{if } b: s_1 \text{ else } s_2](S) & \overset{\text{def}}{=} \text{VARS}(b) \cup \Theta_X[s_1](S) \cup \Theta_X[s_2](S) \\
\Theta_X[\text{while } b: s](S) & \overset{\text{def}}{=} \text{VARS}(b) \cup \Theta_X[s](S) \\
\Theta_X[s_1 \ s_2](S) & \overset{\text{def}}{=} \Theta_X[s_1] \circ \Theta_X[s_2](S)
\end{align*}
\]

\[\begin{array}{c}
ex := v \mid x \mid \text{not } e \mid e \text{ and } e \mid e \text{ or } e \\
s := \text{skip} \mid x := e \mid \text{if } c: s \text{ else } s \mid \text{while } c: s \mid s \ s
\end{array}\]

(expressions) (statements)

\[P \models \mathcal{N}_J \iff \gamma_{\sim}\left(\gamma_X([P]_X)\right) \subseteq \mathcal{N}_J\]

Theorem

sound even for non-terminating programs

passing = True
if not english:
    english = False
if not math:
    passing = False or bonus
if not math:
    passing = False or bonus

\{ bonus, math, english \}
\{ bonus, math \}
\{ bonus \}
\{ passing \}
Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\begin{align*}
\Theta_Q[skip](q) & \overset{\text{def}}{=} q \\
\Theta_Q[x = e](q) & \overset{\text{def}}{=} \text{ASSIGN}[x = e](q) \\
\Theta_Q[\text{if } b: s_1 \text{ else: } s_2](q) & \overset{\text{def}}{=} \text{POP} \circ \text{FILTER}[b] \circ \Theta_Q[s_1] \circ \text{PUSH}(q) \\
& \cup \Theta_Q[\text{while } b: s](q) \overset{\text{def}}{=} \text{IF} \circ \Theta_Q[\text{if } b: s \text{ else: skip}] \\
\Theta_Q[s_1 \ s_2](q) & \overset{\text{def}}{=} \Theta_Q[s_1] \circ \Theta_Q[s_2](q)
\end{align*}
\]

\[
\begin{align*}
\text{passing} & = \text{True} \\
\text{if not } \text{english:} \\
\text{english} & = \text{False} \\
\text{if not } \text{math:} \\
\text{passing} & = \text{False or bonus} \\
\text{if not } \text{math:} \\
\text{passing} & = \text{False or bonus}
\end{align*}
\]

\[
\text{passing} \rightarrow U \quad (\text{any other variable maps to N})
\]
**Syntactic (Non-)Usage**

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\begin{align*}
\theta_Q[\text{skip}](q) & \overset{\text{def}}{=} q \\
\theta_Q[x := e](q) & \overset{\text{def}}{=} \text{ASSIGN}[x := e](q) \\
\theta_Q[\text{if } b: s_1 \text{ else: } s_2](q) & \overset{\text{def}}{=} \text{POP} \circ \text{FILTER}[b] \circ \theta_Q[s_1] \circ \text{PUSH}(q) \\
& \quad \bigcup_{Q} \text{POP} \circ \text{FILTER}[b] \circ \theta_Q[s_2] \circ \text{PUSH}(q) \\
\theta_Q[\text{while } b: s](q) & \overset{\text{def}}{=} \text{if } b: s \text{ else: skip} \\
\theta_Q[s_1 s_2](q) & \overset{\text{def}}{=} \theta_Q[s_1] \circ \theta_Q[s_2](q)
\end{align*}
\]

\[
\begin{align*}
x & \rightarrow U \\
y & \rightarrow S | y \rightarrow U \\
t & \rightarrow N \\
z & \rightarrow N \\
w & \rightarrow O | w \rightarrow U
\end{align*}
\]
**Syntactic (Non-)Usage**

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\begin{align*}
\Theta_Q[\text{skip}](q) & \triangleq q \\
\Theta_Q[x = e](q) & \triangleq \text{ASSIGN}[x = e](q) \\
\Theta_Q[\text{if } b: s_1 \text{ else: } s_2](q) & \triangleq \text{POP} \circ \text{FILTER}[b] \circ \Theta_Q[s_1] \circ \text{PUSH}(q) \\
\Theta_Q[\text{while } b: s](q) & \triangleq \text{if } \Theta_Q[\text{while } b: s]\{q\} \text{ then } \Theta_Q[\text{if } b: s \text{ else: skip}] \text{ else } q \\
\Theta_Q[s_1 s_2](q) & \triangleq \Theta_Q[s_1] \circ \Theta_Q[s_2](q)
\end{align*}
\]

If the assigned variable was used (U or S), it becomes overridden (O) if not also used in the expression being assigned; otherwise, it becomes freshly used (U).

\[
\begin{align*}
\text{passing} & = \text{True} \\
\text{if not } \text{english:} & \\
\text{english} & = \text{False} \\
\text{if not } \text{math:} & \\
\text{passing} & = \text{False or bonus} \\
\text{if not } \text{math:} & \\
\text{passing} & = \text{False or bonus}
\end{align*}
\]

\[
\begin{align*}
\text{bonus} & \rightarrow U, \text{passing} \rightarrow O \mid \text{passing} \rightarrow U \\
\text{passing} & \rightarrow S \mid \text{passing} \rightarrow U \\
\text{passing} & \rightarrow U
\end{align*}
\]
Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\begin{align*}
\Theta_Q[\text{skip}](q) & \overset{\text{def}}{=} q \\
\Theta_Q[x = e](q) & \overset{\text{def}}{=} \text{ASSIGN}[x = e](q) \\
\Theta_Q[\text{if } b: s_1 \text{ else } s_2](q) & \overset{\text{def}}{=} \text{POP} \circ \text{FILTER}[b] \circ \Theta_Q[s_1] \circ \text{PUSH}(q) \\
& \cup_q \text{POP} \circ \text{FILTER}[b] \circ \Theta_Q[s_2] \circ \text{PUSH}(q) \\
\Theta_Q[\text{while } b: s](q) & \overset{\text{def}}{=} \text{IF}_{q} \circ \Theta_Q[\text{if } b: s \text{ else } \text{skip}] \\
\Theta_Q[s_1, s_2](q) & \overset{\text{def}}{=} \Theta_Q[s_1] \circ \Theta_Q[s_2](q)
\end{align*}
\]

pasting = True

if not english:
    english = False

if not math:
    passing = False or bonus

if not math:
    passing = False or bonus

A variable becomes used (U) if it appears in the boolean condition of a statement that uses (U) or modifies (O) another variable.

\[
\begin{align*}
\text{math, bonus, passing} & \rightarrow U \\
\text{bonus} & \rightarrow U, \text{passing} \rightarrow O | \text{passing} \rightarrow U \\
\text{passing} & \rightarrow S | \text{passing} \rightarrow U
\end{align*}
\]
Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\Theta_Q[[\text{skip}]](q) \overset{\text{def}}{=} q
\]
\[
\Theta_Q[[x = e]](q) \overset{\text{def}}{=} \text{assign}[x = e](q)
\]
\[
\Theta_Q[[\text{if } b: s_1 \text{ else: } s_2]](q) \overset{\text{def}}{=} \text{POP} \circ \text{FILTER}[b] \circ \Theta_Q[s_1] \circ \text{PUSH}(q)
\]
\[
\text{if } b: s_1 \text{ else: } s_2
\]
\[
\Theta_Q[[\text{while } b: s]](q) \overset{\text{def}}{=} \text{lfp}_\leq \Theta_Q[[\text{if } b: s \text{ else: skip}]]
\]
\[
\Theta_Q[[s_1 s_2]](q) \overset{\text{def}}{=} \Theta_Q[s_1] \circ \Theta_Q[s_2](q)
\]

\[\begin{align*}
x & \rightarrow U \\
y & \rightarrow S \mid y \rightarrow U \\
t & \rightarrow N \\
z & \rightarrow N \\
w & \rightarrow O \mid w \rightarrow U \\
\end{align*}\]

\[
\boxed{[P]_U}
\]

- **passing** = **True**
- **if not** english:
  - **english** = **False**
- **if not** math:
  - **passing** = **False or bonus**
- **if not** math:
  - **passing** = **False or bonus**

restores the previous value of variables (before increasing the nesting level)
if it has not changed since
Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\begin{align*}
\Theta_Q[\text{skip}](q) & \overset{\text{def}}{=} q \\
\Theta_Q[x = e](q) & \overset{\text{def}}{=} \text{ASSIGN}[x = e](q) \\
\Theta_Q[\text{if } b : s_1 \text{ else: } s_2](q) & \overset{\text{def}}{=} \text{POP} \circ \text{FILTER}[b] \circ \Theta_Q[s_1] \circ \text{PUSH}(q) \\
\Theta_Q[\text{while } b : s](q) & \overset{\text{def}}{=} \text{OPF} \left( \Theta_Q[\text{if } b : s \text{ else: skip}] \right) \\
\Theta_Q[s_1 \ s_2](q) & \overset{\text{def}}{=} \Theta_Q[s_1] \circ \Theta_Q[s_2](q)
\end{align*}
\]

Passing = True

if not english:
    english = False

if not math:
    passing = False or bonus

if not math:
    passing = False or bonus

\[
\begin{align*}
\text{math, bonus, passing} & \rightarrow U \\
\text{math, bonus} & \rightarrow U, \quad \text{passing} \rightarrow O \cup U \quad \text{passing} \rightarrow U \\
\text{bonus} & \rightarrow U, \quad \text{passing} \rightarrow O \quad \text{passing} \rightarrow U \\
\text{passing} & \rightarrow S \quad \text{passing} \rightarrow U \\
\text{passing} & \rightarrow U
\end{align*}
\]
Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\begin{align*}
\Theta_Q[\text{skip}](q) & \overset{\text{def}}{=} q \\
\Theta_Q[x = e](q) & \overset{\text{def}}{=} \text{ASSIGN}[x = e](q) \\
\Theta_Q[\text{if } b: s_1 \text{ else } s_2](q) & \overset{\text{def}}{=} \text{POP} \circ \text{FILTER}[b] \circ \Theta_Q[s_1] \circ \text{PUSH}(q) \\
& \quad \cup \Theta_Q[s_2] \circ \text{PUSH}(q) \\
\Theta_Q[\text{while } b: s](q) & \overset{\text{def}}{=} \text{IF}^Q \Theta_Q[\text{if } b: s \text{ else } \text{skip}] \\
\Theta_Q[s_1 s_2](q) & \overset{\text{def}}{=} \Theta_Q[s_1] \circ \Theta_Q[s_2](q)
\end{align*}
\]

**passing** = True

if not english:
   english = False

if not math:
   passing = False or bonus

if not math:
   passing = False or bonus

math, bonus, passing \rightarrow S \mid math, bonus, passing \rightarrow U
math, bonus, passing \rightarrow U

bonus \rightarrow U, passing \rightarrow O \mid passing \rightarrow U

passing \rightarrow S \mid passing \rightarrow U
passing \rightarrow U
Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\begin{align*}
\Theta_Q[\text{skip}](q) & \overset{\text{def}}{=} q \\
\Theta_Q[x = e](q) & \overset{\text{def}}{=} \text{ASSIGN}[x = e](q) \\
\Theta_Q[\text{if } b: s_1 \text{ else: } s_2](q) & \overset{\text{def}}{=} \text{POP} \circ \text{FILTER}[b] \circ \Theta_Q[s_1] \circ \text{PUSH}(q) \\
& \quad \bigcup \Theta_Q[\text{while } b: s](q) \overset{\text{def}}{=} \text{IF}_b \overset{\text{def}}{=} \Theta_Q[\text{if } b: s \text{ else: skip}] \\
\Theta_Q[s_1 \cdot s_2](q) & \overset{\text{def}}{=} \Theta_Q[\overline{s_1}] \circ \Theta_Q[\overline{s_2}](q)
\end{align*}
\]

- passing = True
- if not english:
  - english = False
- if not math:
  - passing = False or bonus
  - passing = False or bonus
Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\begin{align*}
\theta_Q[\text{skip}](q) & \overset{\text{def}}{=} q \\
\theta_Q[x = e](q) & \overset{\text{def}}{=} \text{ASSIGN}[x = e](q) \\
\theta_Q[\text{if } b: s_1 \text{ else: } s_2](q) & \overset{\text{def}}{=} \text{POP} \circ \text{FILTER}[b] \circ \theta_Q[s_1] \circ \text{PUSH}(q) \\
\theta_Q[\text{while } b: s](q) & \overset{\text{def}}{=} \text{IFP}^{-1} \circ \theta_Q[\text{if } b: s \text{ else: skip}] \\
\theta_Q[s_1, s_2](q) & \overset{\text{def}}{=} \theta_Q[s_1] \circ \theta_Q[s_2](q)
\end{align*}
\]

\[
\begin{align*}
\text{passing} & = \text{True} \\
\text{if not} & \text{ english:} \\
\text{english} & = \text{False} \\
\text{if not} & \text{ math:} \\
\text{passing} & = \text{False or bonus} \\
\text{math, bonus, passing} & \rightarrow U \\
\text{math} & \rightarrow S, \text{ bonus} \rightarrow U, \text{ passing} \rightarrow O \mid \ldots \\
\text{math, bonus, passing} & \rightarrow S | \text{ math, bonus, passing} \rightarrow U \\
\text{math, bonus, passing} & \rightarrow U \\
\text{bonus} & \rightarrow U, \text{ passing} \rightarrow O | \text{ passing} \rightarrow U \\
\text{passing} & \rightarrow S | \text{ passing} \rightarrow U \\
\text{passing} & \rightarrow U
\end{align*}
\]
Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

$\theta_Q[skip](q) \triangleq q$

$\theta_Q[x \leftarrow e](q) \triangleq \text{ASSIGN}[x \leftarrow e](q)$

$\theta_Q[\text{if } b : s_1 \text{ else : } s_2](q) \triangleq \text{POP} \circ \text{FILTER}[b] \circ \theta_Q[s_1] \circ \text{PUSH}(q)$

$\cup_Q \text{POP} \circ \text{FILTER}[b] \circ \theta_Q[s_2] \circ \text{PUSH}(q)$

$\theta_Q[\text{while } b : s](q) \triangleq \text{IF}_Q \theta_Q[\text{if } b : s \text{ else : skip}]$

$\theta_Q[s_1 s_2](q) \triangleq \theta_Q[s_1] \circ \theta_Q[s_2](q)$

- **passing** = **True**

  - if not **english**:
    
    - **english** = **False**
    
    - if not **math**:
      
      - passing = **False** or **bonus**
      
      - if not **math**:
        
        - passing = **False** or **bonus**

- **math**, **bonus**, **passing** → **S** | **math**, **bonus**, **passing** → **U**

- **math** → **S**, **bonus** → **U**, **passing** → **O** | ...
### Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\begin{align*}
\text{passing} &= \text{True} \\
\text{if not} \text{ english:} \\
\quad &\text{english} = \text{False} \\
\text{if not} \text{ math:} \\
\quad &\text{passing} = \text{False or} \text{ bonus} \\
\text{if not} \text{ math:} \\
\quad &\text{passing} = \text{False or} \text{ bonus} \\
\end{align*}
\]
### Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

\[
\begin{align*}
\text{passing} &= \text{True} \\
\text{if not english:} \\
&\quad \text{english} = \text{False} \\
\text{if not math:} \\
&\quad \text{passing} = \text{False or bonus} \\
\text{if not math:} \\
&\quad \text{passing} = \text{False or bonus} \\
\end{align*}
\]

- \(x \rightarrow U\)
- \(y \rightarrow S \mid y \rightarrow U\)
- \(t \rightarrow N\)
- \(z \rightarrow N\)
- \(w \rightarrow O \mid w \rightarrow U\)

\[
\Theta_Q[\text{skip}](q) \overset{\text{def}}{=} q
\]

\[
\Theta_Q[x := e](q) \overset{\text{def}}{=} \text{ASSIGN}[x := e](q)
\]

\[
\Theta_Q[\text{if } b: s_1 \text{ else: } s_2](q) \overset{\text{def}}{=} \text{POP} \circ \text{FILTER}[b] \circ \Theta_Q[s_1] \circ \text{PUSH}(q)
\]

\[
\Theta_Q[\text{while } b: s](q) \overset{\text{def}}{=} \text{if } \neg b \text{ then } s \text{ else: } \text{skip} \text{ end } \Theta_Q[s](q)
\]

\[
\Theta_Q[s_1 s_2](q) \overset{\text{def}}{=} \Theta_Q[s_1] \circ \Theta_Q[s_2](q)
\]

\[\mathbb{P}_U\]
Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

The input variables `english` and `science` are definitely not used by the program.

- `math, bonus → U`, `passing → O`
- `math, bonus, passing → U`
- `math, bonus, passing → S | math, bonus, passing → U`
- `math, bonus, passing → S | math, bonus, passing → U`
- `math, bonus, passing → U`
- `english = False`
- `passing = True`
- `if not english:
  english = False`
- `if not math:
  passing = False or bonus`
- `if not math:
  passing = False or bonus`
- `θ_Q[if b: s1 else: s2](q) ≜ POP ◦ FILTER[b] ◦ θ_Q[s1] ◦ PUSH(q)`
- `θ_Q[while b: s](q) ≜ interp ◦ θ_Q[if b: s else: skip]`
- `θ_Q[s_1 s_2](q) ≜ θ_Q[s_1] ◦ θ_Q[s_2](q)`

\[ [P]_U \]
Syntactic (Non-)Usage

- **U**: used in the current scope (or an inner scope)
- **S**: used in an outer scope
- **O**: used in an outer scope and overridden in the current scope
- **N**: not used

Theorem

\[ P \vdash N^*_j \iff \gamma_\prec(\gamma_Q([P]_Q)) \subseteq N^*_j \]
Piecewise Syntactic (Non-)Usage

\begin{align*}
\text{grades} &= \text{list}(\text{map}(\text{int}, \text{input}().\text{split}())) \\
\text{count} &= 0 \\
i &= 1 \\
\text{while} \ i < \text{len}(\text{grades}): \\
\quad \text{if} \ \text{grades}[i] < 4: \\
\quad \quad \text{count} = \text{count} + 1 \\
\quad i = i + 1 \\
\text{if} \ 2 \ast \text{count} < \text{len}(\text{grades}): \\
\quad \text{passing} = \text{True} \\
\text{else}: \\
\quad \text{passing} = \text{False} \\
\text{print}(\text{passing})
\end{align*}

ERROR: 1 SHOULD BE 0
Piecewise Syntactic (Non-)Usage

\[ e ::= v \mid x \mid \text{not } e \mid e \text{ and } e \mid e \text{ or } e \]
\[ s ::= \text{skip} \mid x = e \mid \text{if } c : s \text{ else: } s \mid \text{while } c : s \mid s s \]

grades = \text{list(map(int, input().split()))}

count = 0

i = 1

while i < len(grades):
    if grades[i] < 4:
        count = count + 1

    i = i + 1

if 2 * count < len(grades):
    passing = True
else:
    passing = False

print(passing)
Piecewise Syntactic (Non-)Usage

\[
e ::= v \mid x \mid \text{not } e \mid e \land e \mid e \lor e
\]

\[
s ::= \text{skip} \mid x = e \mid \text{if } e : s \text{ else: } s \mid \text{while } e : s
\]

\[
\begin{align*}
\text{grades} &= \text{list(map(int, input().split()))} \\
\text{count} &= 0 \\
i &= 1
\end{align*}
\]

\[
\text{while } i < \text{len(grades)}:
\begin{align*}
\text{if } \text{grades}[i] < 4: \\
\quad \text{count} &= \text{count} + 1
\end{align*}
\]

\[
i &= i + 1
\]

\[
\begin{align*}
\text{if } 2 \times \text{count} < \text{len(grades)}: \\
\quad \text{passing} &= \text{True}
\end{align*}
\]

\[
\text{else:} \\
\quad \text{passing} &= \text{False}
\]

\[
\text{print} \left( \text{passing} \right)
\]
Piecewise Syntactic (Non-)Usage

\[ e ::= v \mid x \mid \text{not} \ e \mid \text{and} \ e \mid \text{or} \ e \]
\[ s ::= \text{skip} \mid x = e \mid \text{if} \ e \mid s \text{ else } s \mid \text{while} \ e \mid s \mid s \]

```
grades = list(map(int, input().split()))
count = 0
i = 1
while i < len(grades):
    if grades[i] < 4:
        count = count + 1
    i = i + 1
if 2 * count < len(grades):
    passing = True
else:
    passing = False
print(passing)
```

ERROR: 1 SHOULD BE 0

grades \rightarrow \{0\} N \{i\}? U \{i+1\}? U \{len(grades)\}?

grades \rightarrow \{0\} N \{i\}? U \{i+1\}? S \{i+2\}? S \{len(grades)\}? | ...

grades \rightarrow \{0\} N \{i\}? U \{i+1\}? U \{len(grades)\}?

grades \rightarrow \{0\} N \{i+1\}? S \{i+2\}? S \{len(grades)\}? | ...

grades \rightarrow \{0\} N \{i+1\}? S \{i+2\}? S \{len(grades)\}? | ...

grades \rightarrow \{0\} N \{i+1\}? S \{i+2\}? S \{len(grades)\}? | ...

grades \rightarrow \{0\} N \{i\}? U \{i+1\}? U \{len(grades)\}?

grades \rightarrow \{0\} N \{len(grades)\}?
Piecewise Syntactic (Non-)Usage

```
e ::= v \mid x \mid \text{not } e \mid e \text{ and } e \mid e \text{ or } e
s ::= \text{skip} \mid x = e \mid \text{if } e : s \text{ else: } s \mid \text{while } e : s \mid s s

grades = \text{list(map(int, input().split()))}
count = 0
i = 1
while i < \text{len}(grades):
    if grades[i] < 4:
        count = count + 1
    i = i + 1
if 2 \times \text{count} < \text{len}(grades):
    passing = True
else:
    passing = False
\text{print}(passing)
```

**ERROR:** 1 SHOULD BE 0

- grades[0] is definitely not used by the program

grades[0] \rightarrow [0] N \{1\} U \{2\} U \{\text{len}(grades)\}?

grades \rightarrow [0] N \{i\} U \{i+1\} U \{\text{len}(grades)\}?

grades \rightarrow [0] N \{i\} U \{i+1\} S \{i+2\} S \{\text{len}(grades)\}? | …

grades \rightarrow [0] N \{i\} U \{i+1\} S \{i+2\} S \{\text{len}(grades)\}? | … | …

grades \rightarrow [0] N \{i\} S \{i+1\} S \{\text{len}(grades)\}? | …

grades \rightarrow [0] N \{\text{len}(grades)\}?
Implicit Assumptions on Data
What Programs Want
Automatic Inference of Input Data Specifications

Caterina Urban\textsuperscript{1,2}
\textsuperscript{1} INRIA, Paris, France
\textsuperscript{2} DIENS, École Normale Supérieure, CNRS, PSL University, Paris, France
caterina.urban@inria.fr

Abstract. Nowadays, as machine-learned software quickly permeates our society, we are becoming increasingly vulnerable to programming errors in the data pre-processing or training software, as well as errors in the data itself. In this paper, we propose a static shape analysis framework for input data of data-processing programs. Our analysis automatically infers necessary conditions on the structure and values of the data read by a data-processing program. Our framework builds on a family of underlying abstract domains, extended to indirectly reason about the input data rather than simply reasoning about the program variables. The choice of these abstract domain is a parameter of the analysis. We describe various instances built from existing abstract domains. The proposed approach is implemented in an open-source static analyzer for \textsc{python} programs. We demonstrate its potential on a number of representative examples.

1 Introduction

Due to the availability of vast amounts of data and corresponding tremendous advances in machine learning, computer software is nowadays an ever increasing presence in every aspect of our society. As we rely more and more on machine-learned software, we become increasingly vulnerable to programming errors but (in contrast to traditional software) also errors in the data used for training.

In general, before software training, the data goes through long pre-processing pipelines\textsuperscript{3}. Errors can be missed, or even introduced, at any stage of these pipelines. This is even more true when data pre-processing stages are disregarded as single-use glue code and, for this reason, are poorly tested, let alone statically analyzed or verified. Moreover, this kind of code is often written in a rush and is highly dependent on the data (e.g., the use of magic constants is not uncommon) All this together, greatly increases the likelihood for errors to be noticed extremely late in the pipeline (which entails a more or less important waste of time), or more dangerously, to remain completely unnoticed.

\textsuperscript{3}https://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html


