Static Analysis for Data Science

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

Caterina Urban

November 7th, 2022

Year 2022-2023

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

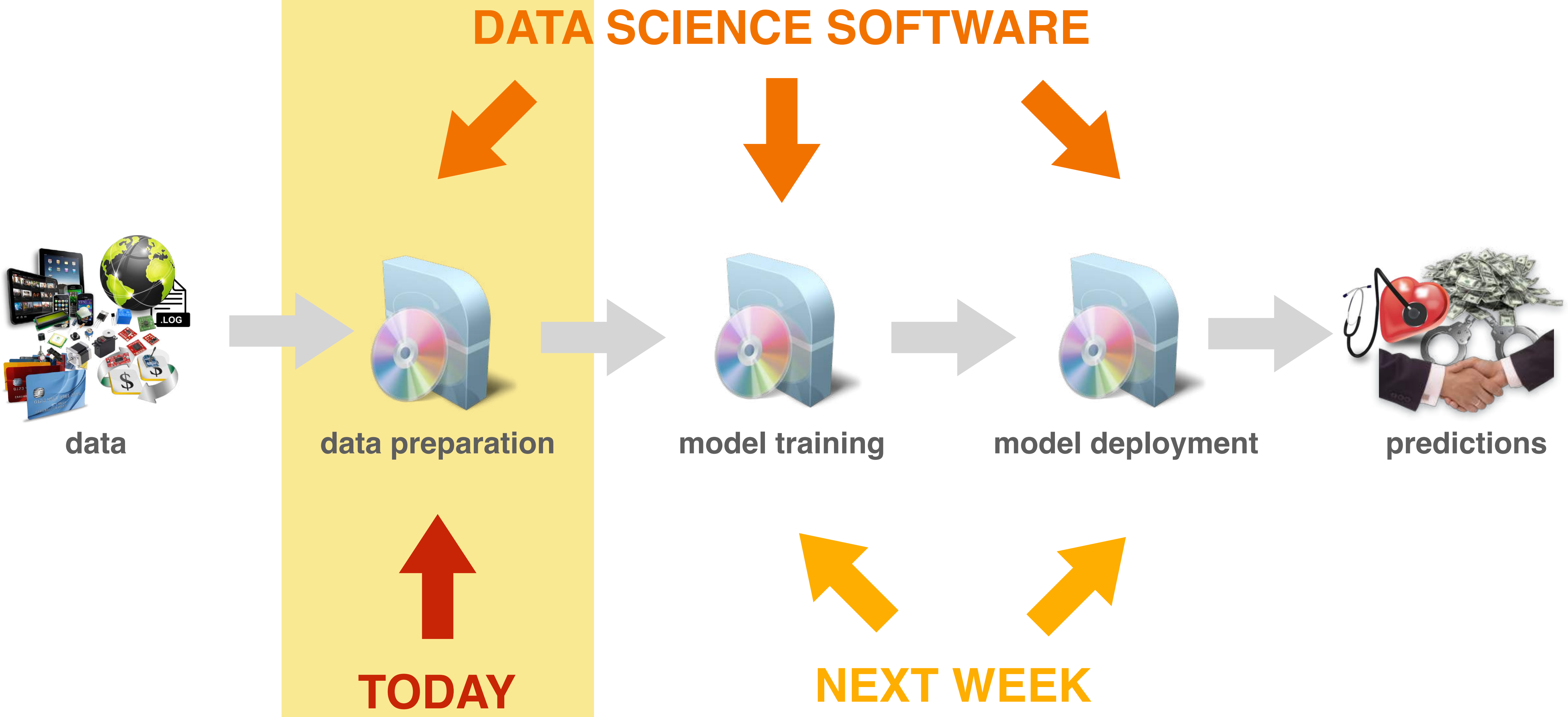
pharmaceutical
- predictive models
- patient selection

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 SCIENCE SOFTWARE

data

data preparation

model training

model deployment

predictions

TODAY

NEXT WEEK

Ubiquitous Programming Errors

data science means programming

Excel

R

Python

Julia

programming means programming errors

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


pharmaceutical


energy


finance


health care

Anomalously Unused Data

The Reinhart-Rog

American Economic Review: Papers & Proceedings 100 (May 2010): 573–578
http://www.aeaweb.org/articles.php?doi=10.1257/aer.100.2.573

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 (government) debt to search for a systemic relationship between high public debt levels, growth and inflation.[1] 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 maelstrom, especially in the epicenter countries. This should not be surprising, given the experience of earlier severe financial crises.[2] Outsized 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.

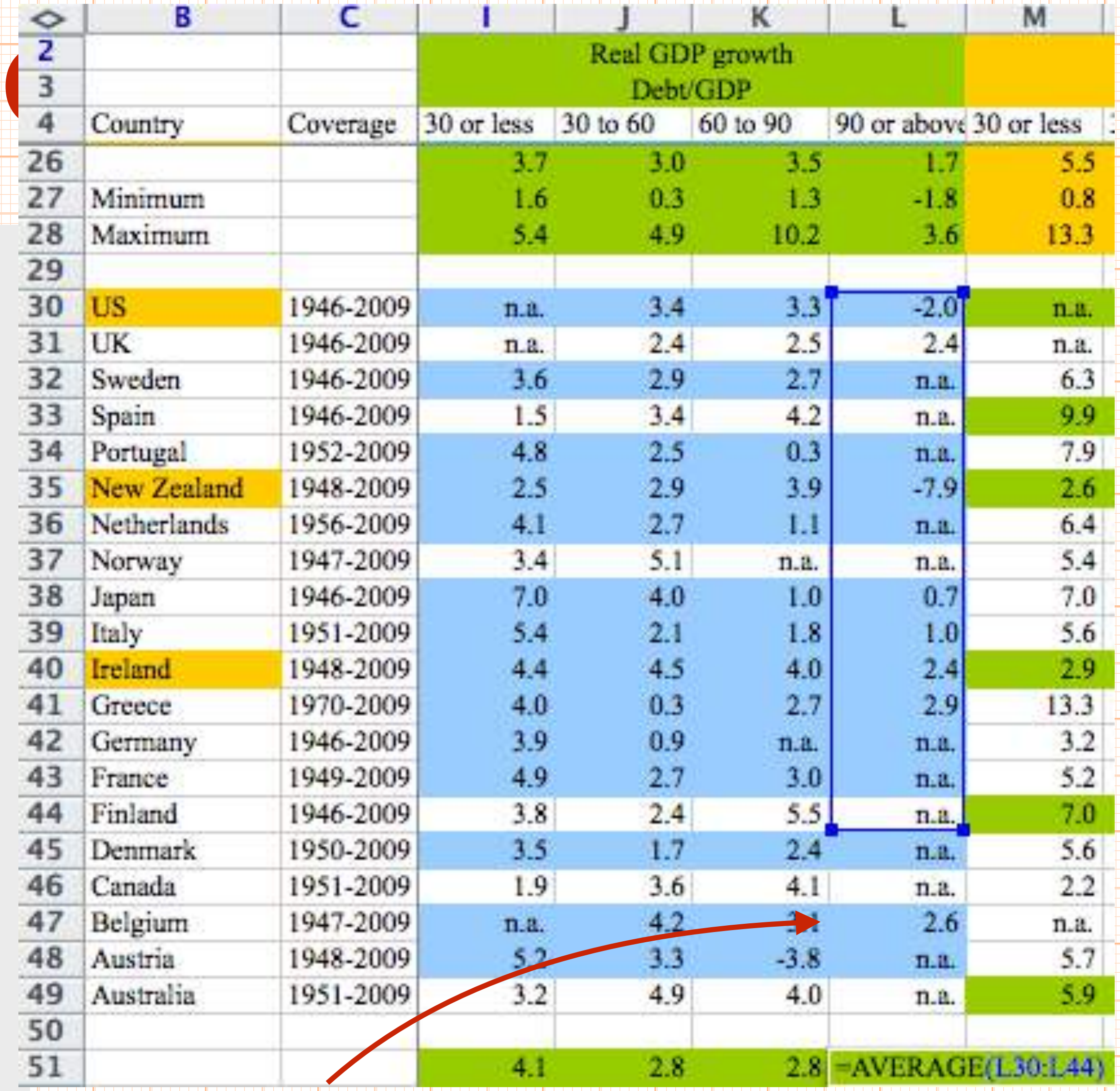
The focus of this paper is on the longer term macroeconomic implications of much higher public and external debt. The final section, how-

---

*Reinhart: Department of Economics, 4115 Tydings Hall, University of Maryland, College Park, MD 20742 (e-mail: creinhar@umd.edu); Rogoff: Economics Department, 216 Littauer Center, Harvard University, Cambridge MA 02138–3001 (e-mail: krogoff@harvard.edu). The authors would like to thank Olivier Jeanne and Vincent R. Reinhart for helpful comments.

[1] In this paper "public debt" refers to gross central government debt. "Domestic public debt" is government debt issued under domestic legal jurisdiction. Public debt does not include debts carrying a government guarantee. Total gross external debt includes the external debts of *all* branches of government as well as private debt that is issued by domestic private entities under a foreign jurisdiction.

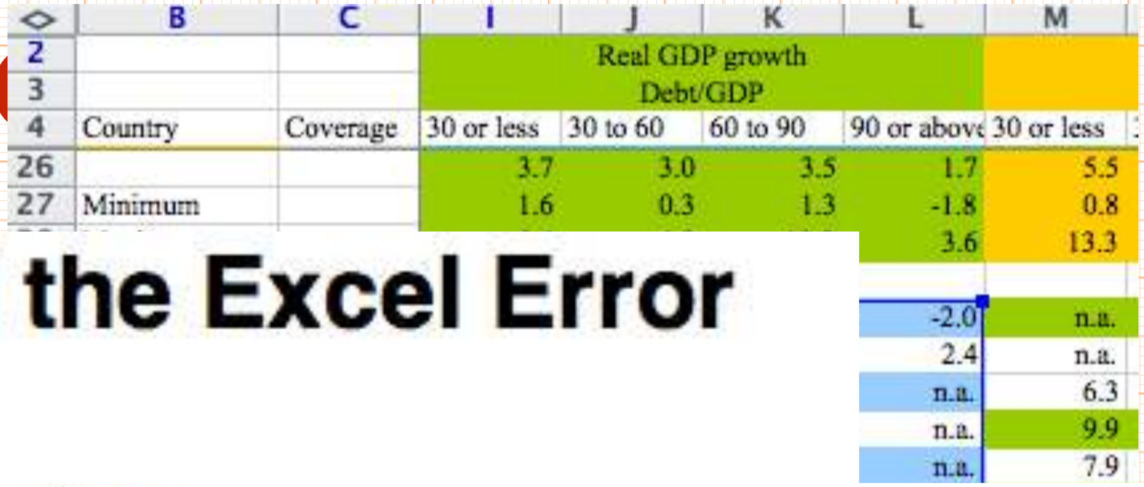
[2] Reinhart and Rogoff (2009a, b) demonstrate that the aftermath of a deep financial crisis typically involves a protracted period of macroeconomic adjustment, particularly in employment and housing prices. On average, public



◇	B	C	I	J	K	L	M
2			Real GDP growth				
3			Debt/GDP				
4	Country	Coverage	30 or less	30 to 60	60 to 90	90 or above	30 or less
26			3.7	3.0	3.5	1.7	5.5
27	Minimum		1.6	0.3	1.3	-1.8	0.8
28	Maximum		5.4	4.9	10.2	3.6	13.3
29							
30	US	1946-2009	n.a.	3.4	3.3	-2.0	n.a.
31	UK	1946-2009	n.a.	2.4	2.5	2.4	n.a.
32	Sweden	1946-2009	3.6	2.9	2.7	n.a.	6.3
33	Spain	1946-2009	1.5	3.4	4.2	n.a.	9.9
34	Portugal	1952-2009	4.8	2.5	0.3	n.a.	7.9
35	New Zealand	1948-2009	2.5	2.9	3.9	-7.9	2.6
36	Netherlands	1956-2009	4.1	2.7	1.1	n.a.	6.4
37	Norway	1947-2009	3.4	5.1	n.a.	n.a.	5.4
38	Japan	1946-2009	7.0	4.0	1.0	0.7	7.0
39	Italy	1951-2009	5.4	2.1	1.8	1.0	5.6
40	Ireland	1948-2009	4.4	4.5	4.0	2.4	2.9
41	Greece	1970-2009	4.0	0.3	2.7	2.9	13.3
42	Germany	1946-2009	3.9	0.9	n.a.	n.a.	3.2
43	France	1949-2009	4.9	2.7	3.0	n.a.	5.2
44	Finland	1946-2009	3.8	2.4	5.5	n.a.	7.0
45	Denmark	1950-2009	3.5	1.7	2.4	n.a.	5.6
46	Canada	1951-2009	1.9	3.6	4.1	n.a.	2.2
47	Belgium	1947-2009	n.a.	4.2	2.1	2.6	n.a.
48	Austria	1948-2009	5.2	3.3	-3.8	n.a.	5.7
49	Australia	1951-2009	3.2	4.9	4.0	n.a.	5.9
50							
51			4.1	2.8	2.8	=AVERAGE(L30:L44)	

data excluded from the analysis

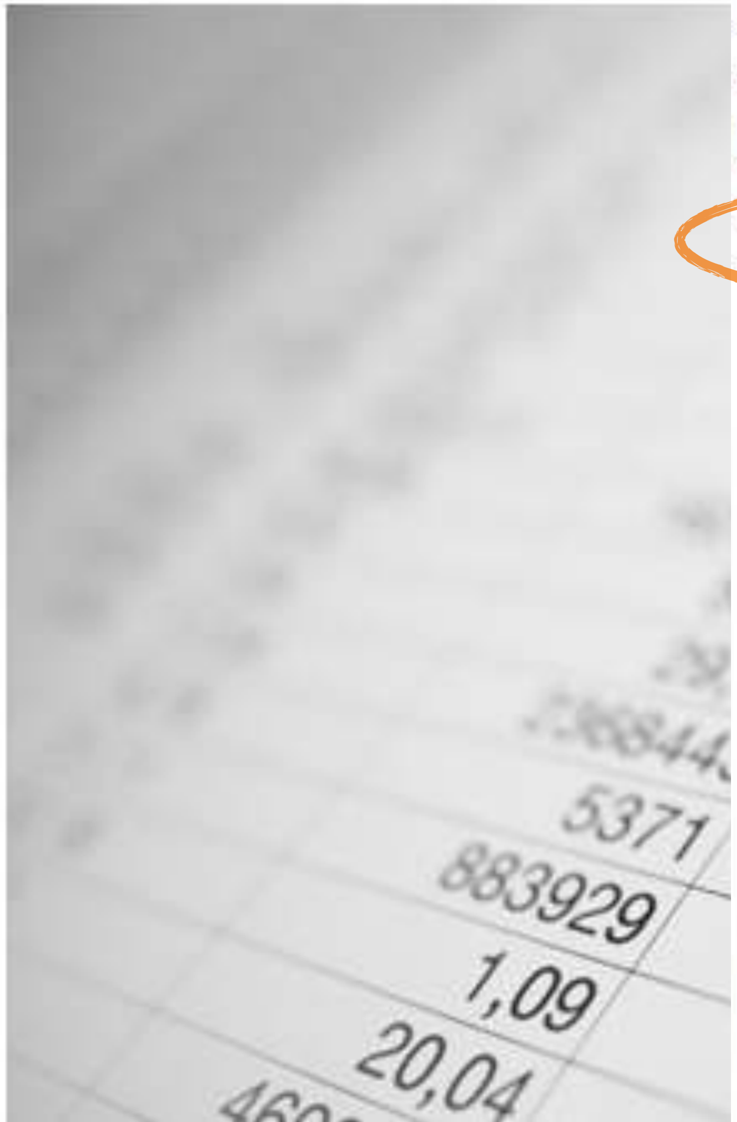
The Reinhart-Rog



	B	C	I	J	K	L	M
2			Real GDP growth				
3			Debt/GDP				
4	Country	Coverage	30 or less	30 to 60	60 to 90	90 or abov	30 or less
26			3.7	3.0	3.5	1.7	5.5
27	Minimum		1.6	0.3	1.3	-1.8	0.8
						3.6	13.3
						-2.0	n.a.
						2.4	n.a.
						n.a.	6.3
						n.a.	9.9
						n.a.	7.9

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

By Peter Coy | April 18, 2013



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?

Enlarge This Image

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

FACEBOOK
TWITTER
GOOGLE+
SAVE
EMAIL
SHARE
PRINT
REPRINTS

England Covid-19 Cases Error

SCIENCE \ US & WORLD \ TECH

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

NEWS

Check for updates

The BMJ

Cite this as: *BMJ* 2020;371:m3891

http://dx.doi.org/10.1136/bmj.m3891

Published: 06 October 2020

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, ading this deadly virus at a time when

data and furthermore have issued guidance on validation and risk management for these products if they are to be used in such a safety critical manner."

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 ing in their areas. Routine regular testing in high high transmission areas would in 24 hours.

Example

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

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

print(passing)
```

INPUT VARIABLES

ERROR: english SHOULD BE passing

ERROR: math SHOULD BE science

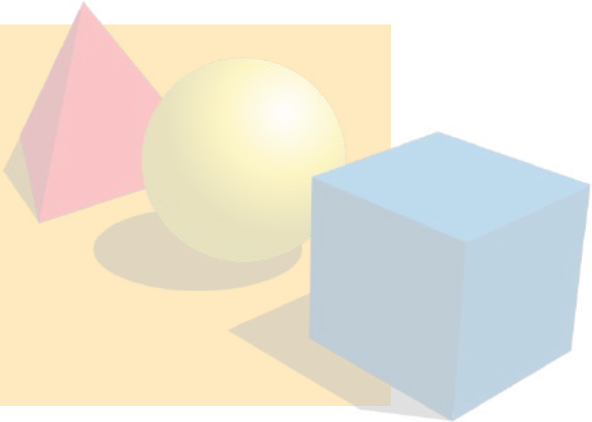
OUTPUT VARIABLES

the input variables english and science are unused

Unused Data Analysis

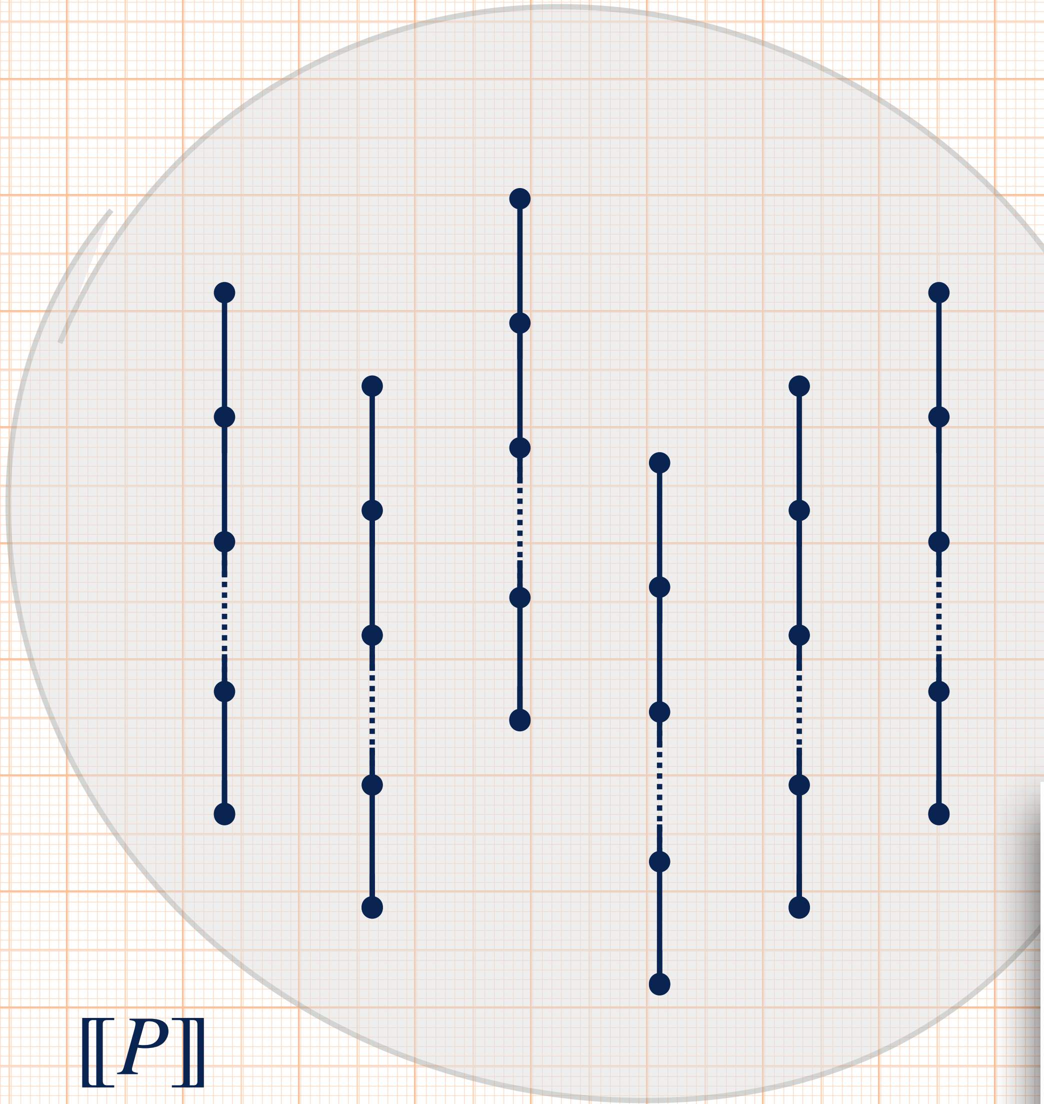
practical tools
targeting specific programs

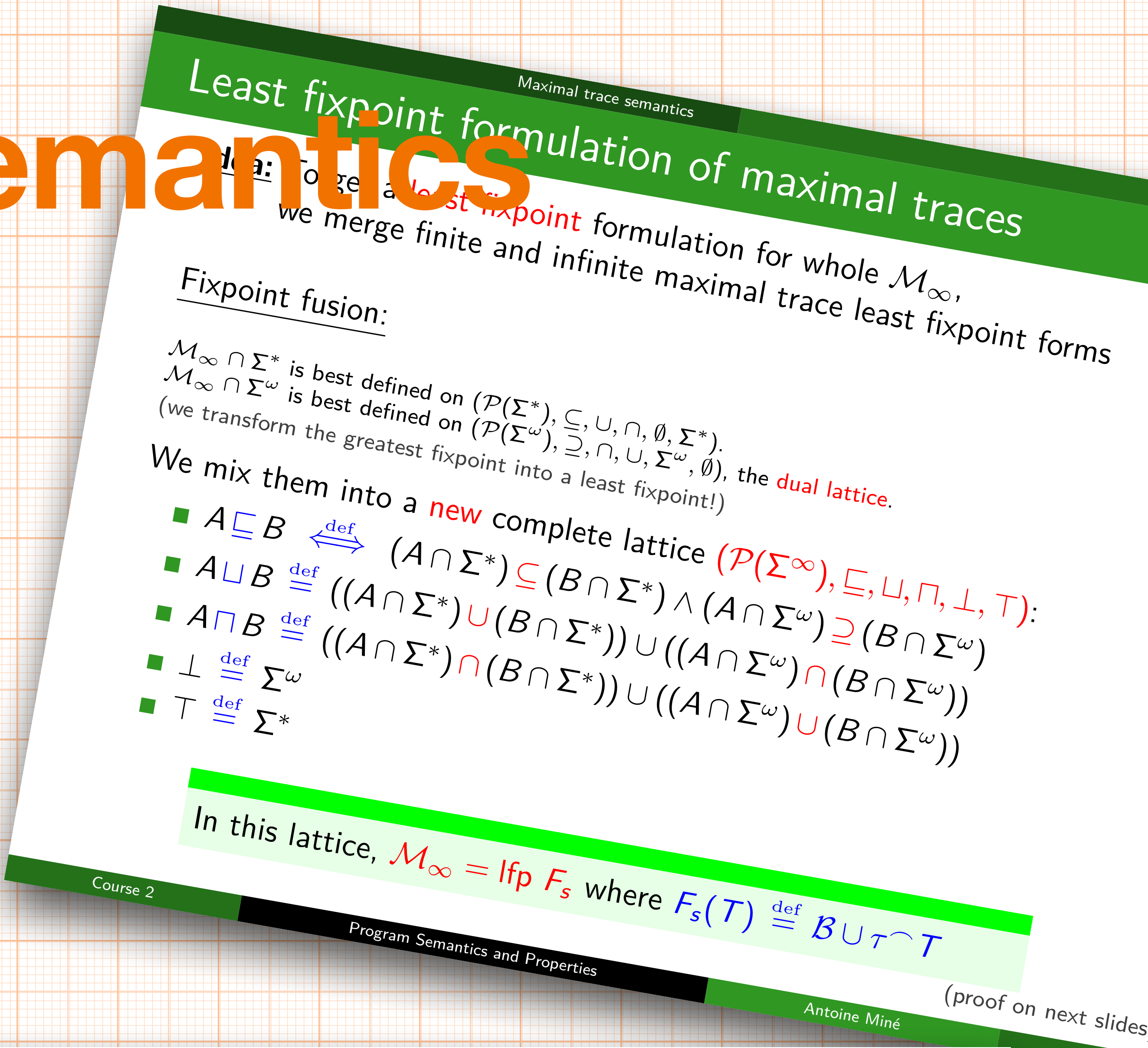
algorithmic approaches
to decide program properties



mathematical models
of the program behavior

Maximal Trace Semantics




Maximal trace semantics

Least fixpoint formulation of maximal traces

a: To ge a least fixpoint formulation for whole $\mathcal{M}_\infty$,
we merge finite and infinite maximal trace least fixpoint forms

Fixpoint fusion:

$\mathcal{M}_\infty \cap \Sigma^*$ is best defined on $(\mathcal{P}(\Sigma^*), \subseteq, \cup, \cap, \emptyset, \Sigma^*)$.
$\mathcal{M}_\infty \cap \Sigma^\omega$ is best defined on $(\mathcal{P}(\Sigma^\omega), \supseteq, \cap, \cup, \Sigma^\omega, \emptyset)$, the dual lattice.
(we transform the greatest fixpoint into a least fixpoint!)

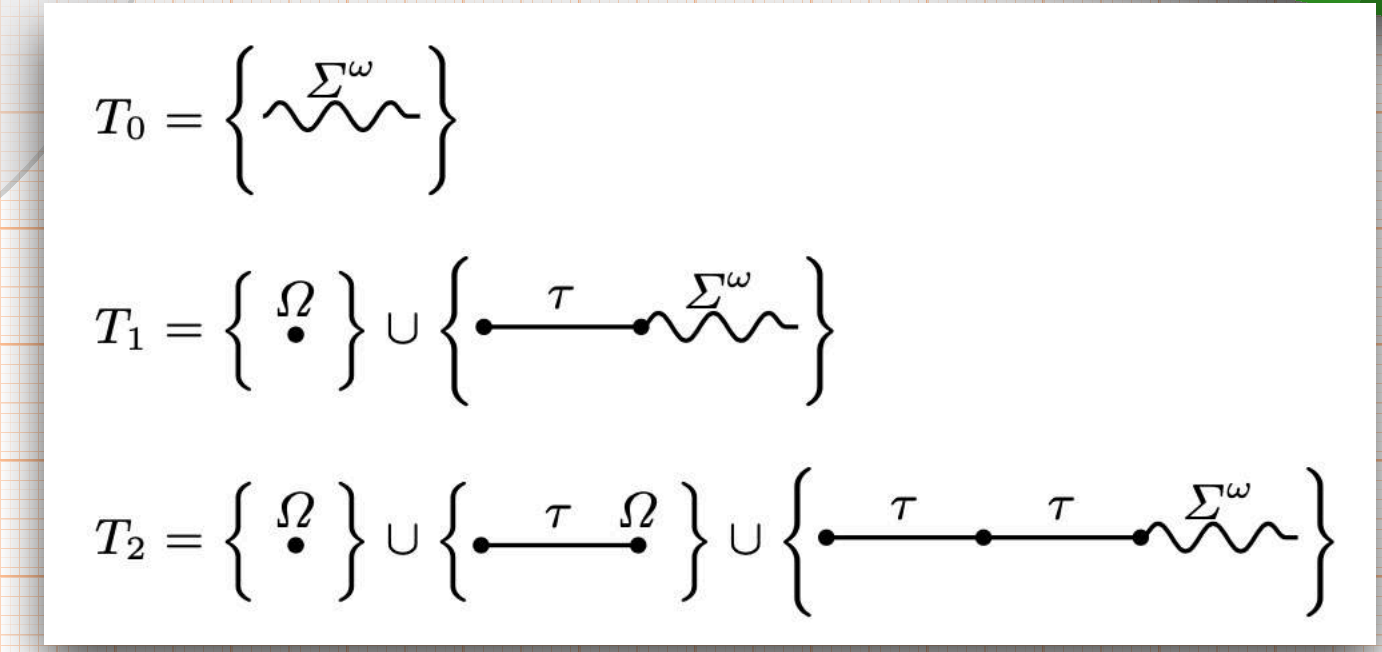
We mix them into a new complete lattice $(\mathcal{P}(\Sigma^\infty), \sqsubseteq, \sqcup, \sqcap, \bot, \top)$:

- $A \sqsubseteq B \overset{\text{def}}{\iff} (A \cap \Sigma^*) \subseteq (B \cap \Sigma^*) \wedge (A \cap \Sigma^\omega) \supseteq (B \cap \Sigma^\omega)$
- $A \sqcup B \overset{\text{def}}{=} ((A \cap \Sigma^*) \cup (B \cap \Sigma^*)) \cup ((A \cap \Sigma^\omega) \cap (B \cap \Sigma^\omega))$
- $A \sqcap B \overset{\text{def}}{=} ((A \cap \Sigma^*) \cap (B \cap \Sigma^*)) \cup ((A \cap \Sigma^\omega) \cup (B \cap \Sigma^\omega))$
- $\bot \overset{\text{def}}{=} \Sigma^\omega$
- $\top \overset{\text{def}}{=} \Sigma^*$

In this lattice, $\mathcal{M}_\infty = \text{lfp } F_s$ where $F_s(T) \overset{\text{def}}{=} \mathcal{B} \cup \tau \frown T$

(proof on next slides

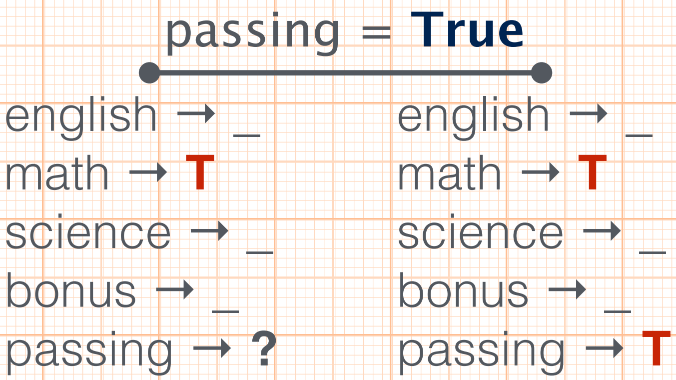
Course 2 | Program Semantics and Properties | Antoine Miné | p. 76 / 98



Maximal Trace Semantics

$[\![P]\!]$

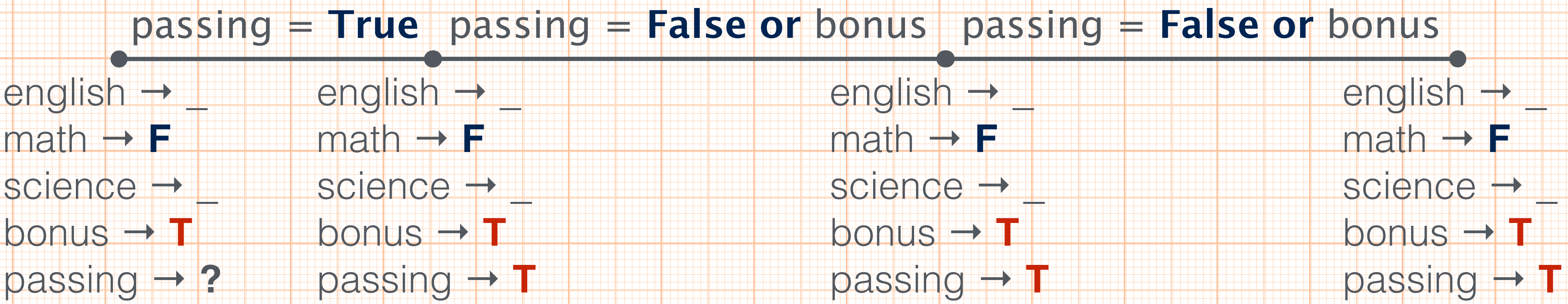
```
passing = True
if not english:
    english = False
if not math:
    passing = False or bonus
if not math:
    passing = False or bonus
```



passing = True

english → _
math → **T**
science → _
bonus → _
passing → **?**

english → _
math → **T**
science → _
bonus → _
passing → **T**



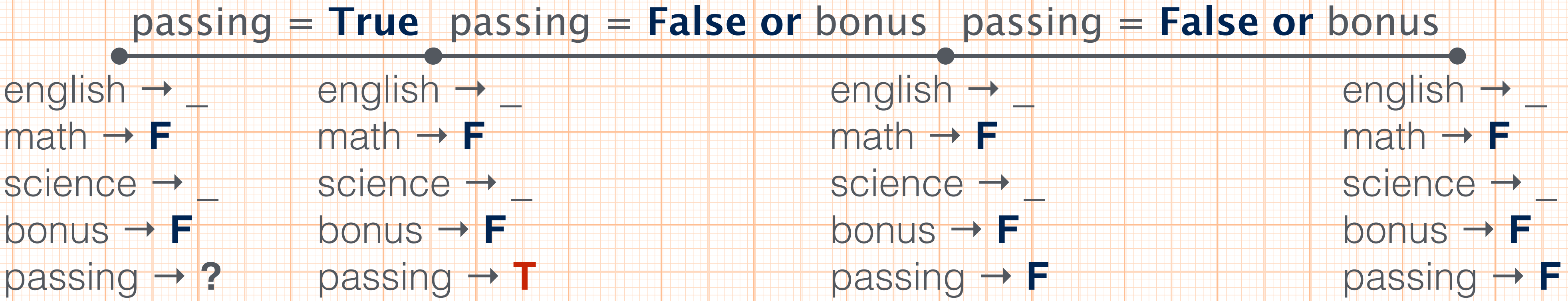
passing = **True** | passing = **False or** bonus | passing = **False or** bonus

english → _
math → **F**
science → _
bonus → **T**
passing → **?**

english → _
math → **F**
science → _
bonus → **T**
passing → **T**

english → _
math → **F**
science → _
bonus → **T**
passing → **T**

english → _
math → **F**
science → _
bonus → **T**
passing → **T**



passing = **True** | passing = **False or** bonus | passing = **False or** bonus

english → _
math → **F**
science → _
bonus → **F**
passing → **?**

english → _
math → **F**
science → _
bonus → **F**
passing → **T**

english → _
math → **F**
science → _
bonus → **F**
passing → **F**

english → _
math → **F**
science → _
bonus → **F**
passing → **F**

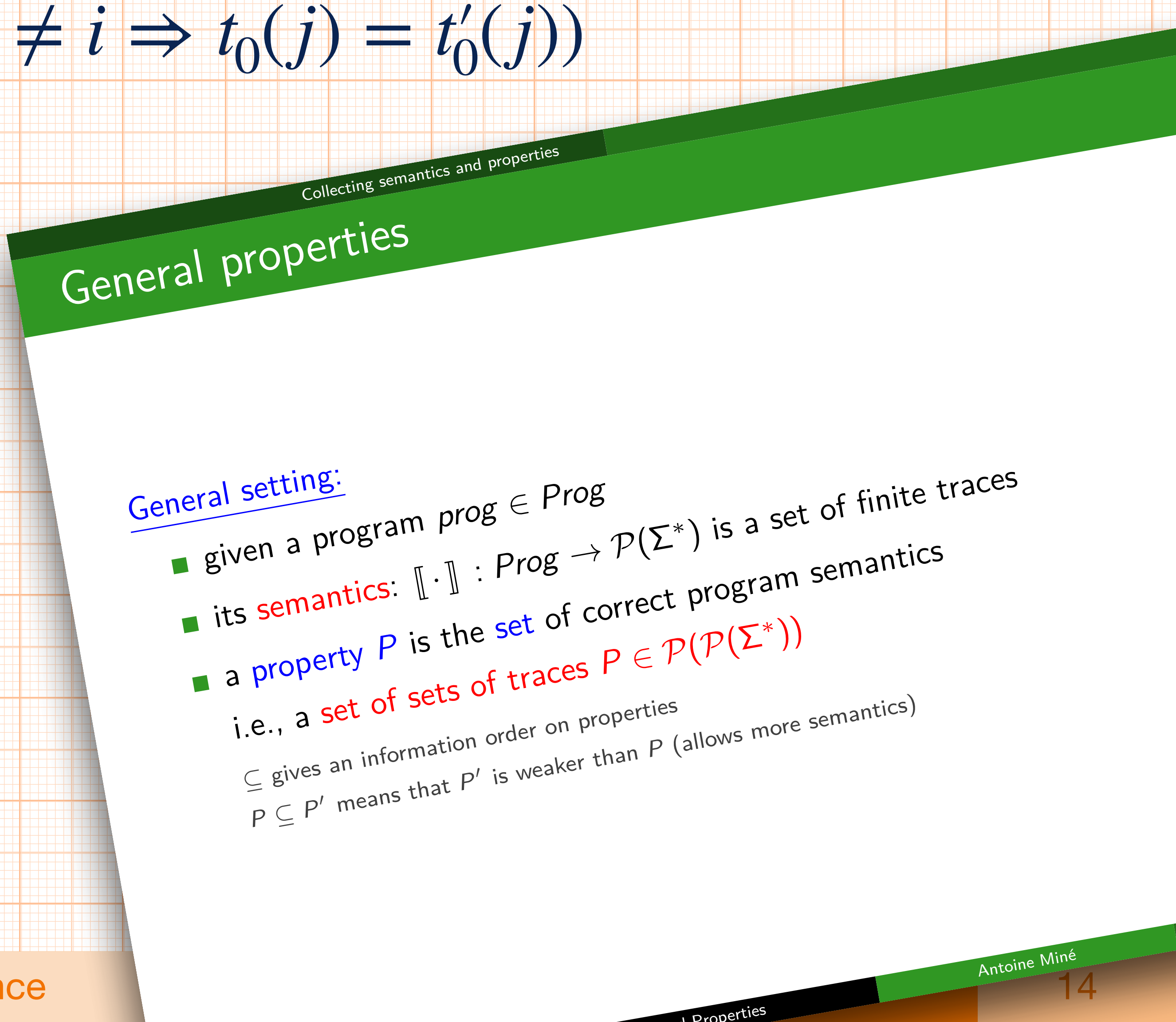
Input Data (Non-)Usage

$$\mathscr{N}_J \overset{\text{def}}{=} \{\llbracket P \rrbracket \in \mathscr{P}(\Sigma^{+\infty}) \mid \forall i \in J \subseteq I_P \colon \text{UNUSED}_i(\llbracket P \rrbracket)\}$$

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

$$\begin{aligned} \text{UNUSED}_i(\llbracket M \rrbracket) \overset{\text{def}}{=} \; & \forall t \in \llbracket P \rrbracket, v \in \mathscr{V} : t_0(i) \neq v \Rightarrow \exists t' \in \llbracket P \rrbracket : \\ & (\forall 0 \leq j \leq |I_P| : j \neq i \Rightarrow t_0(j) = t'_0(j)) \\ & \wedge t'_0(i) = v \\ & \wedge t_\omega = t'_\omega \end{aligned}$$

Intuitively: **any possible program outcome** is possible **from any value of the input** variable $x_{0,i}$



Collecting semantics and properties

General properties

General setting:

- given a program $prog \in Prog$
- its semantics: $\llbracket \cdot \rrbracket : Prog \to \mathcal{P}(\Sigma^*)$ is a set of finite traces
- a property P is the set of correct program semantics

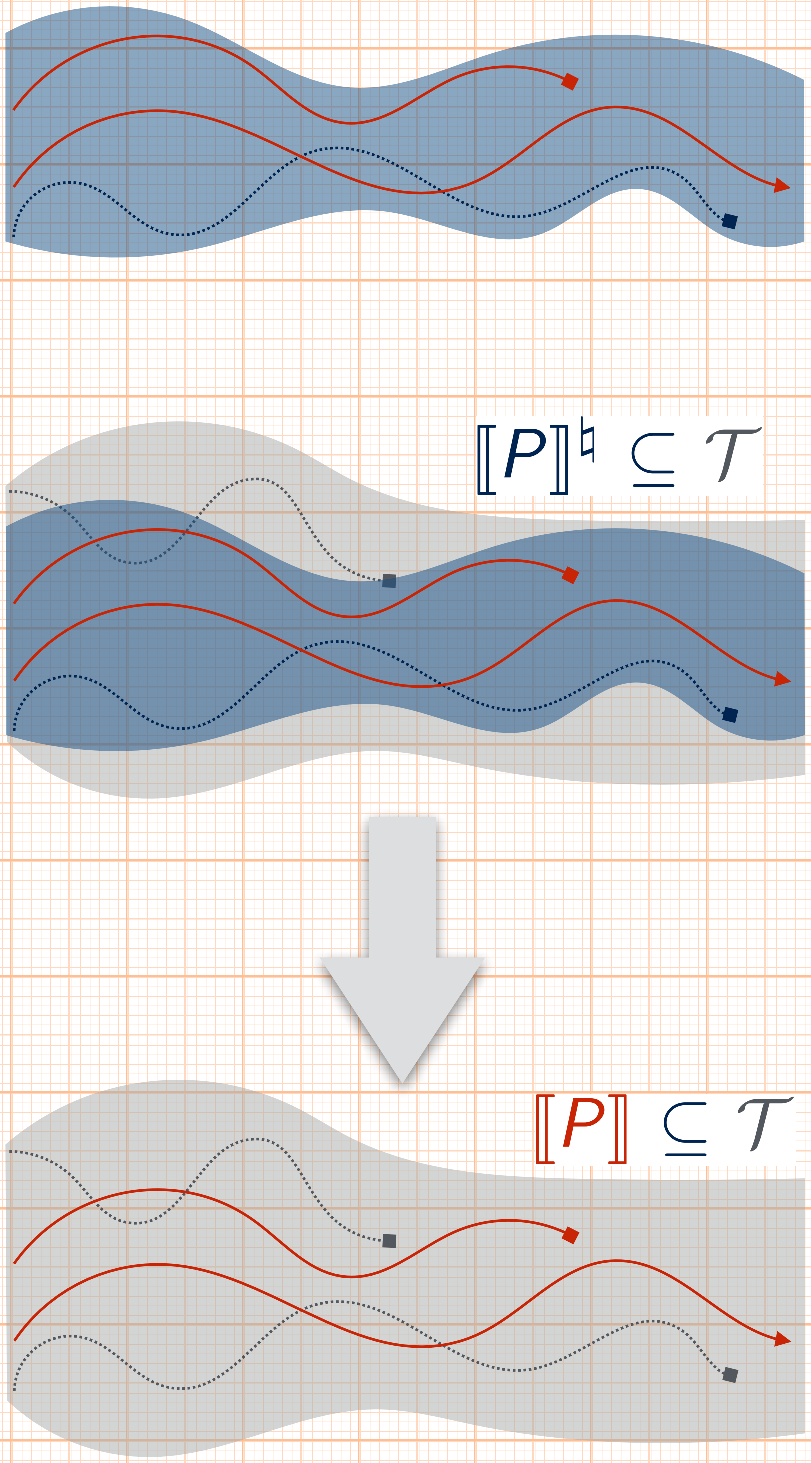
  i.e., a set of sets of traces $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$

  $\subseteq$ gives an information order on properties

  $P \subseteq P'$ means that P' is weaker than P (allows more semantics)

Antoine Miné

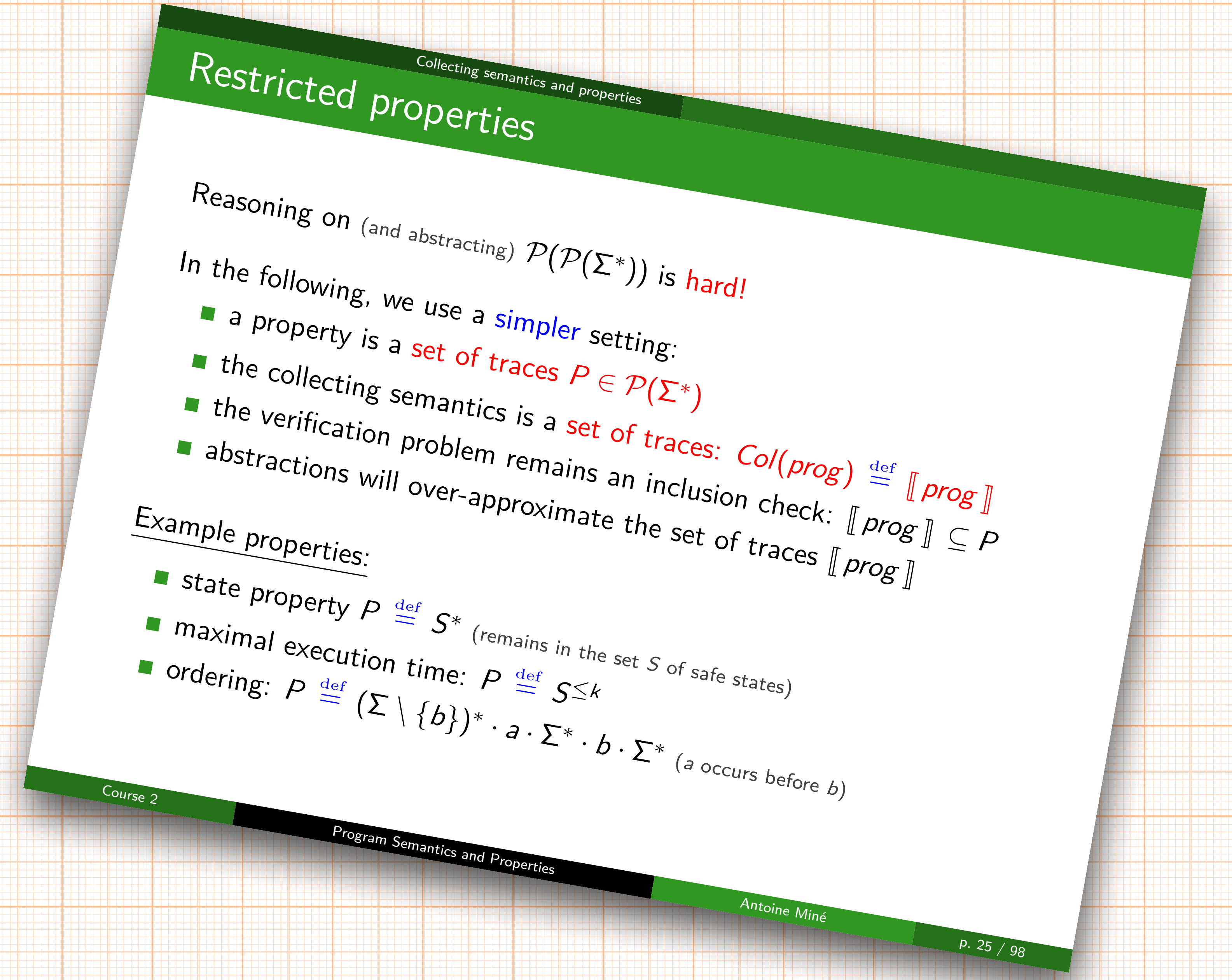
Trace Properties



$[\![P]\!] \subseteq [\![P]\!]^\natural$

$[\![P]\!]^\natural \subseteq \mathcal{T}$

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



Restricted properties

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

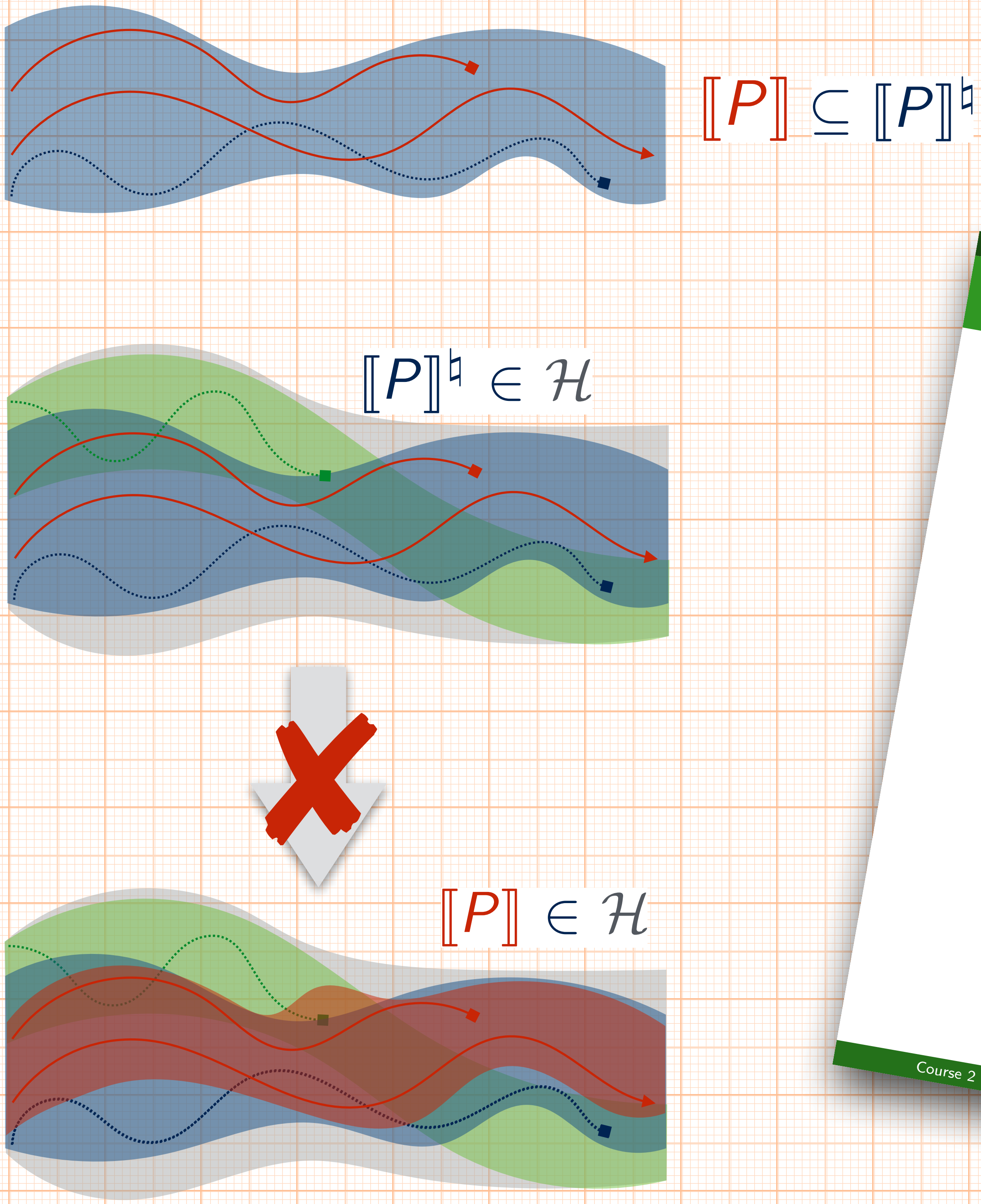
In the following, we use a simpler setting:

- a property is a set of traces $P \in \mathcal{P}(\Sigma^*)$
- the collecting semantics is a set of traces: $Col(prog) \stackrel{\text{def}}{=} [\![\, prog \,]\!]$
- the verification problem remains an inclusion check: $[\![\, prog \,]\!] \subseteq P$
- abstractions will over-approximate the set of traces $[\![\, prog \,]\!]$

Example properties:

- state property $P \stackrel{\text{def}}{=} S^*$ (remains in the set S of safe states)
- maximal execution time: $P \stackrel{\text{def}}{=} S^{\leq k}$
- ordering: $P \stackrel{\text{def}}{=} (\Sigma \setminus \{b\})^* \cdot a \cdot \Sigma^* \cdot b \cdot \Sigma^*$ (a occurs before b)

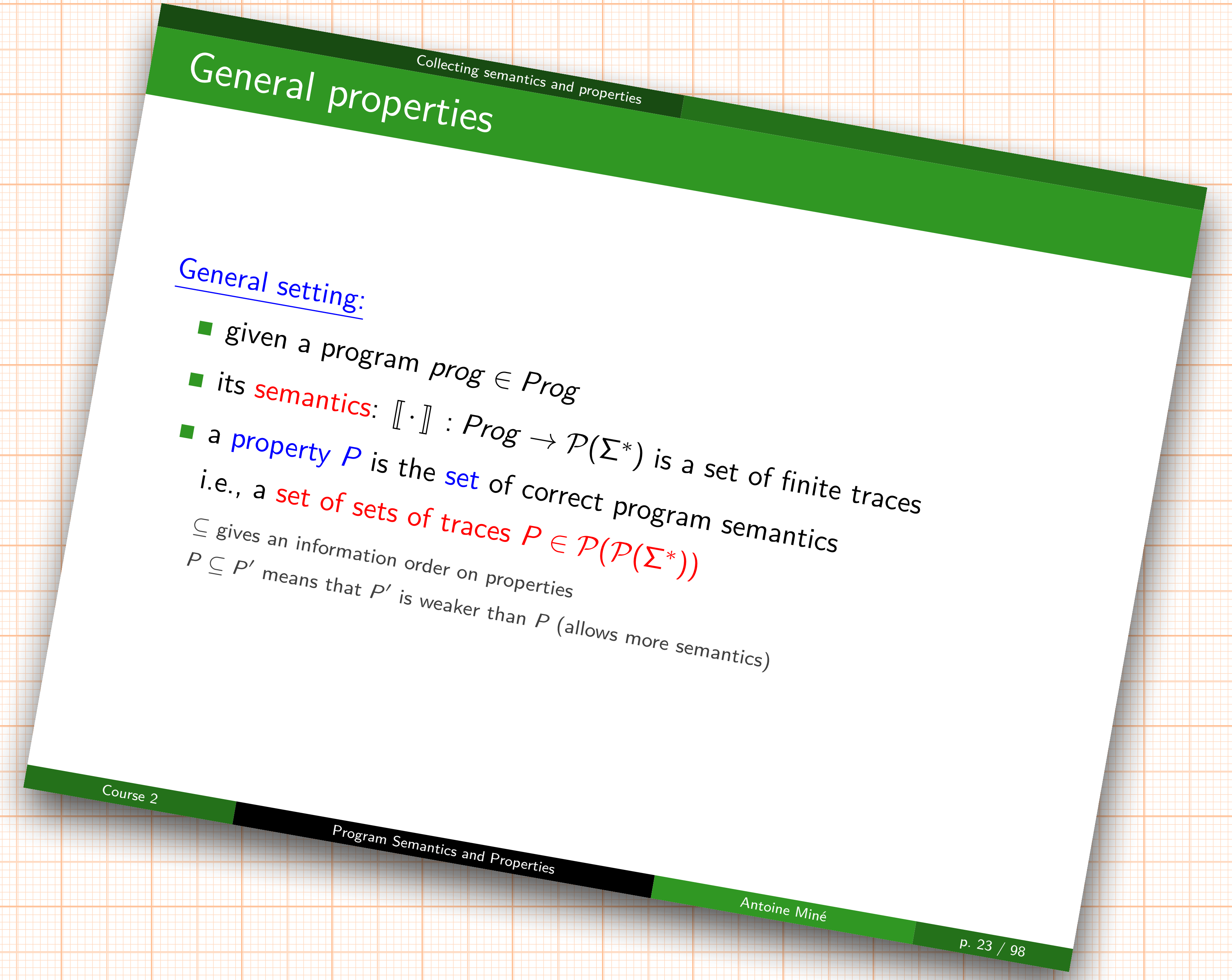
Program Properties



$[\![P]\!] \subseteq [\![P]\!]^\natural$

$[\![P]\!]^\natural \in \mathcal{H}$

$[\![P]\!] \in \mathcal{H}$



Collecting semantics and properties

General properties

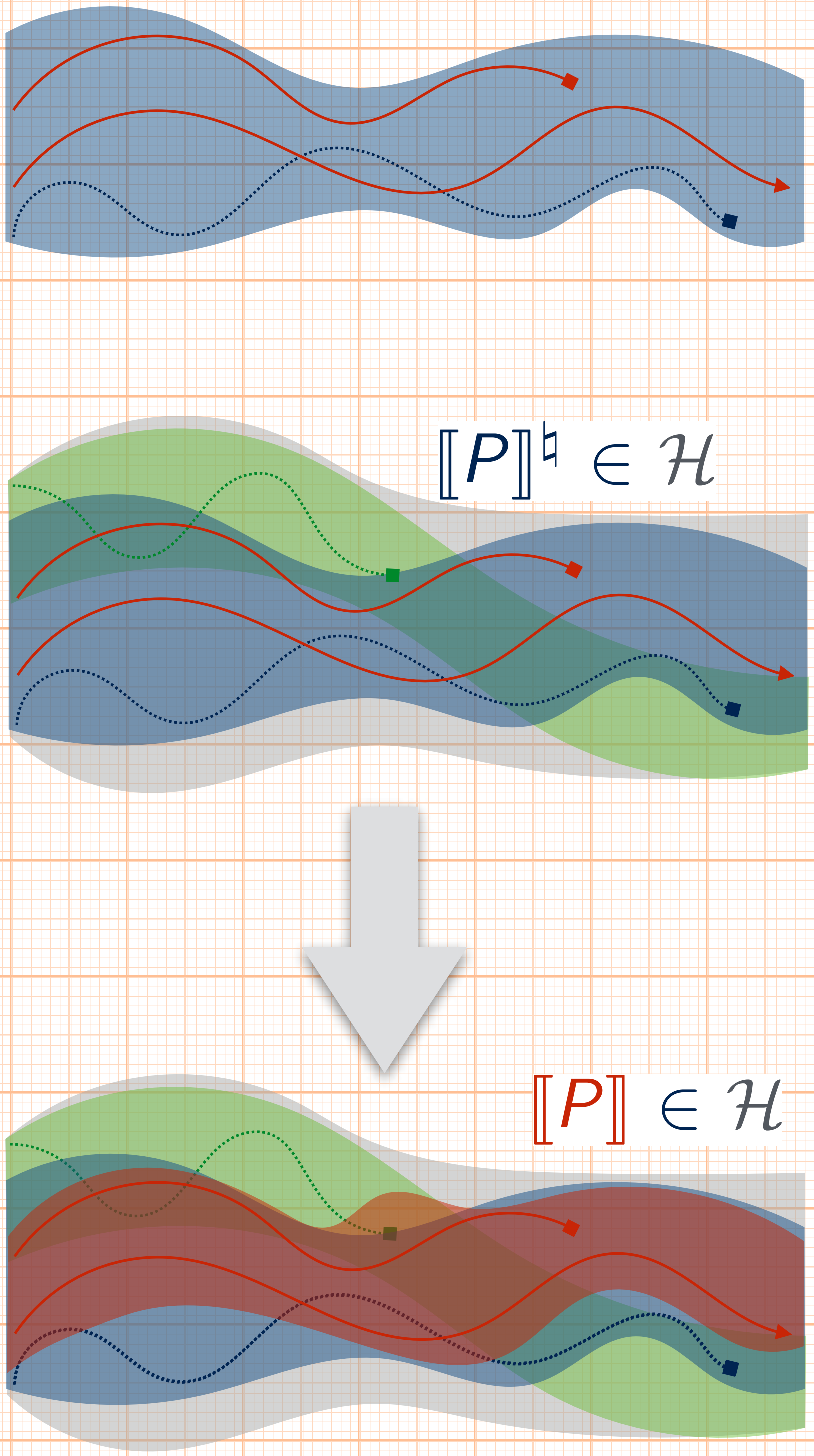
General setting:

- given a program $prog \in Prog$
- its semantics: $[\![\cdot]\!] : Prog \to \mathcal{P}(\Sigma^*)$ is a set of finite traces
- a property P is the set of correct program semantics
  i.e., a set of sets of traces $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$

$\subseteq$ gives an information order on properties
$P \subseteq P'$ means that P' is weaker than P (allows more semantics)

Course 2 | Program Semantics and Properties | Antoine Miné | p. 23 / 98

Subset-Closed Properties



$$[\![P]\!] \subseteq [\![P]\!]^\natural$$

$$[\![P]\!]^\natural \in \mathcal{H}$$

$$[\![P]\!] \in \mathcal{H}$$

Program Properties

$[\![P]\!] \subseteq [\![P]\!]^\natural$

$[\![P]\!]^\natural \in \mathcal{H}$

$[\![P]\!] \in \mathcal{H}$

General properties

Collecting semantics and properties

General setting:

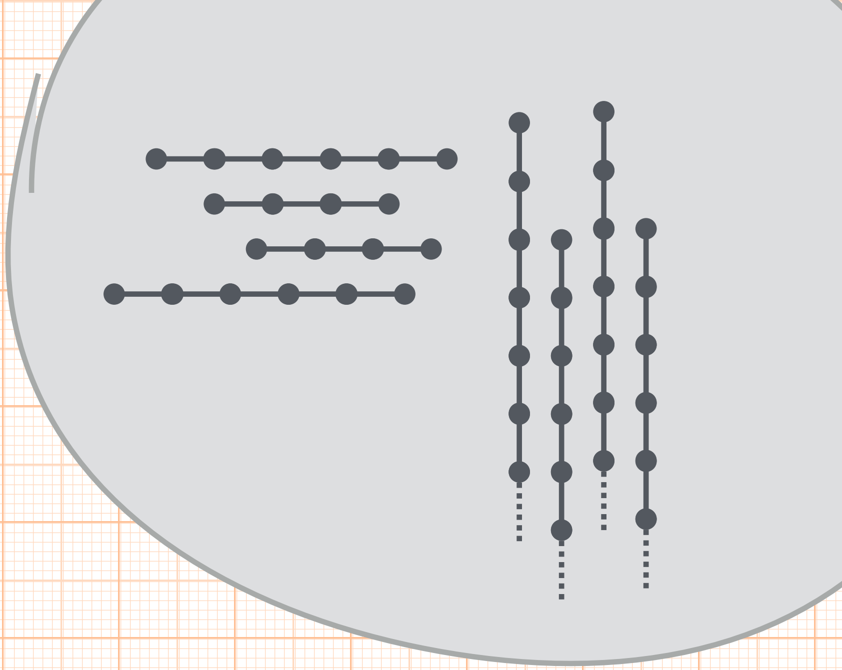
- given a program $prog \in Prog$
- its semantics: $[\![\cdot]\!] : Prog \to \mathcal{P}(\Sigma^*)$ is a set of finite traces
- a property P is the set of correct program semantics
i.e., a set of sets of traces $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$

$\subseteq$ gives an information order on properties
$P \subseteq P'$ means that P' is weaker than P (allows more semantics)

Course 2 | Program Semantics and Properties | Antoine Miné | p. 23 / 98

Lesson 7 | Static Analysis for Data Science | Caterina Urban | 15

Input Data (Non-)Usage



$[\![P]\!]$

```
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 → **?**

english → _
math → **T**
science → _
bonus → _
passing → **T**

passing = **True** passing = **False or** bonus passing = **False or** bonus

english → _
math → **F**
science → _
bonus → **T**
passing → **?**

english → _
math → **F**
science → _
bonus → **T**
passing → **T**

english → _
math → **F**
science → _
bonus → **T**
passing → **T**

english → _
math → **F**
science → _
bonus → **T**
passing → **T**

passing = **True** passing = **False or** bonus passing = **False or** bonus

english → _
math → **F**
science → _
bonus → **F**
passing → **?**

english → _
math → **F**
science → _
bonus → **F**
passing → **T**

english → _
math → **F**
science → _
bonus → **F**
passing → **F**

english → _
math → **F**
science → _
bonus → **F**
passing → **F**

Input Data (Non-)Usage

$$\mathscr{N}_J \overset{\text{def}}{=} \{[\![P]\!] \in \mathscr{P}(\Sigma^{+\infty}) \mid \text{UNUSED}_J([\![P]\!])\}$$

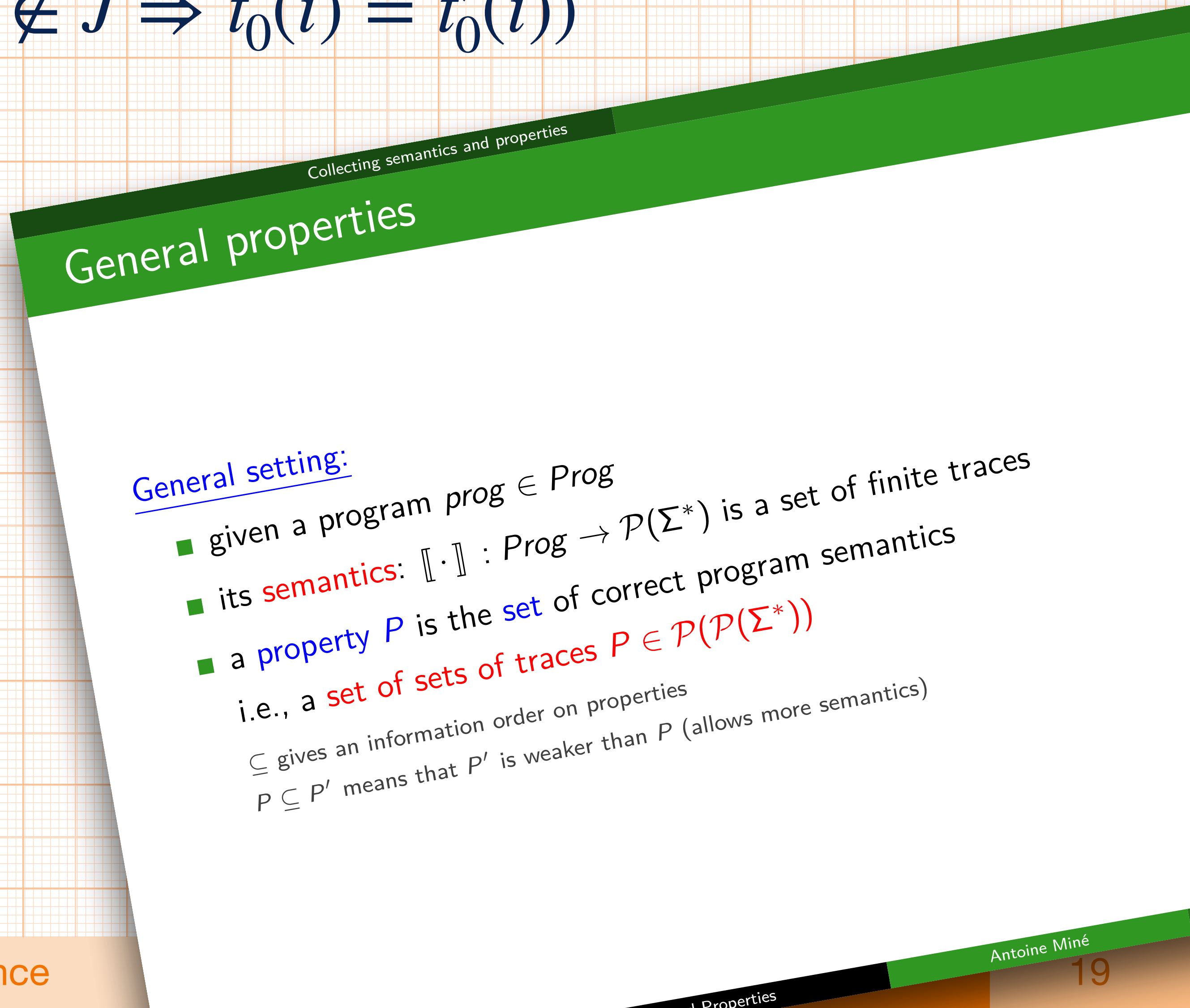
$\mathscr{N}_J$ is the set of all programs P (or, rather, their semantics $[\![P]\!]$) that **do not use** the value of the input variables in $J \subseteq I_P$

$$\begin{aligned}\text{UNUSED}_J([\![M]\!]) \overset{\text{def}}{=} \; & \forall t \in [\![P]\!], V \in \mathscr{V} : t_0(J) \neq V \Rightarrow \exists t' \in [\![P]\!] : \\ & (\forall 0 \leq i \leq |I_P| : i \notin J \Rightarrow t_0(i) = t'_0(i)) \\ & \wedge t'_0(J) = V \\ & \wedge t_\omega = t'_\omega\end{aligned}$$

Intuitively: **any possible program outcome** is possible **from any value of the input** variables in J

Theorem

$$P \models \mathscr{N}_J \Leftrightarrow \{[\![P]\!]\} \subseteq \mathscr{N}_J$$



Collecting semantics and properties

General properties

General setting:

- given a program $prog \in Prog$
- its semantics: $[\![\cdot]\!] : Prog \to \mathcal{P}(\Sigma^*)$ is a set of finite traces
- a property P is the set of correct program semantics

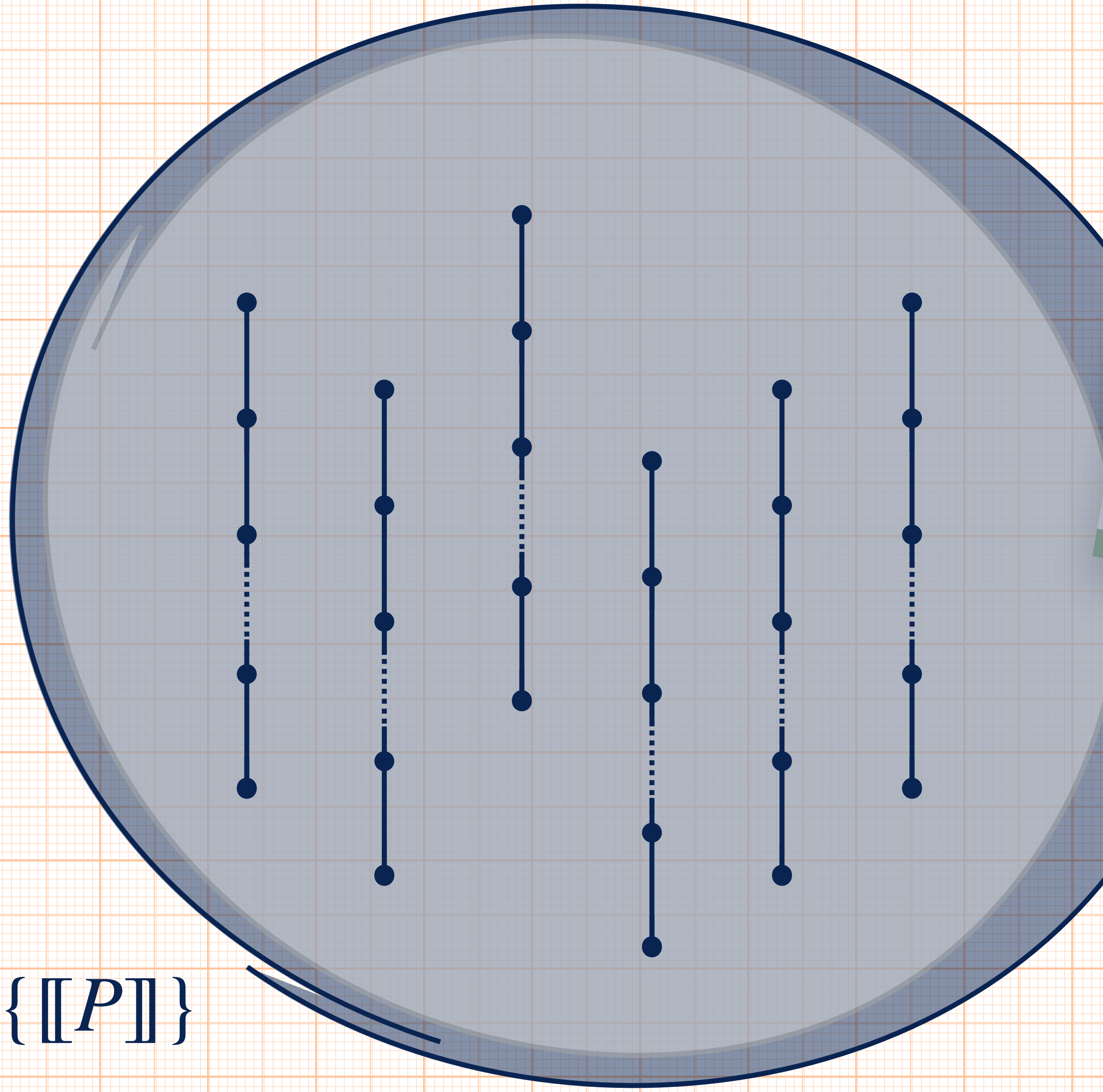
  i.e., a set of sets of traces $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$

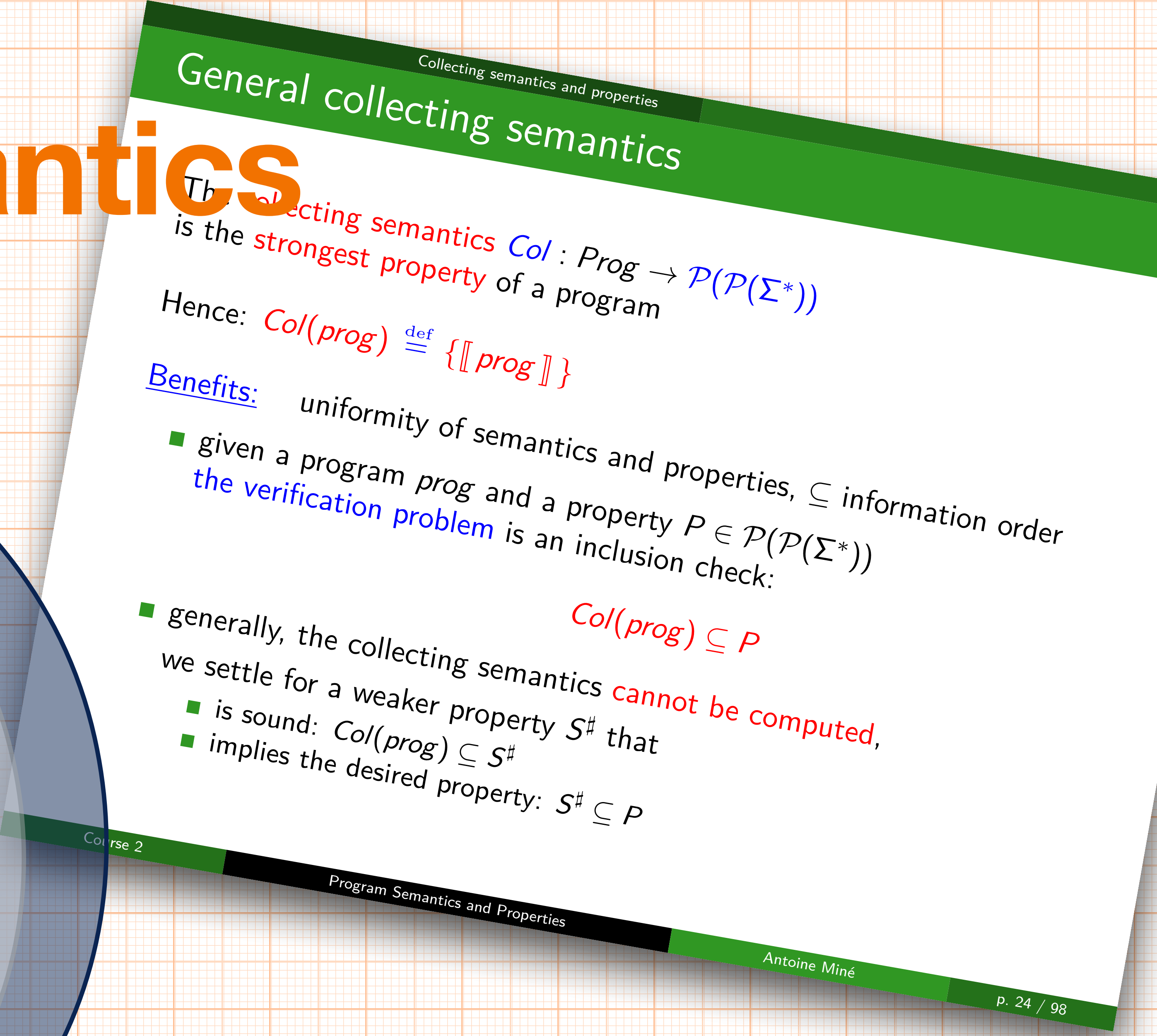
$\subseteq$ gives an information order on properties

$P \subseteq P'$ means that P' is weaker than P (allows more semantics)

Antoine Miné

Collecting Semantics




Collecting semantics and properties

General collecting semantics

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

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

Benefits: uniformity of semantics and properties, $\subseteq$ information order

- given a program *prog* and a property $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$ the verification problem is an inclusion check:

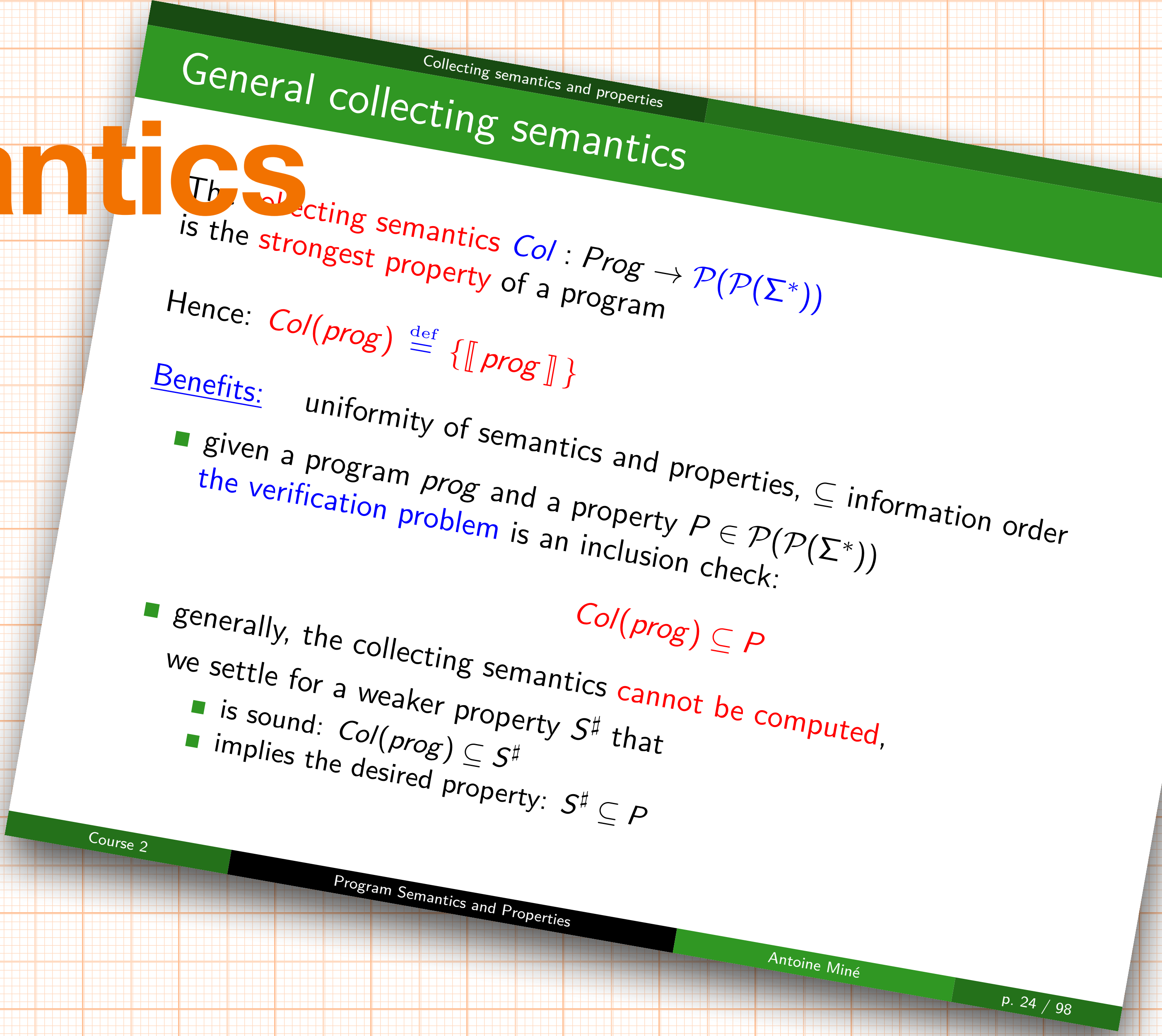
$$Col(prog) \subseteq P$$

- generally, the collecting semantics cannot be computed, we settle for a weaker property $S^\sharp$ that
  - is sound: $Col(prog) \subseteq S^\sharp$
  - implies the desired property: $S^\sharp \subseteq P$

Course 2 | Program Semantics and Properties | Antoine Miné | p. 24 / 98

Collecting Semantics

Intuition



Collecting semantics and properties

General collecting semantics

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

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

Benefits: uniformity of semantics and properties, $\subseteq$ information order

- given a program *prog* and a property $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$
  the verification problem is an inclusion check:

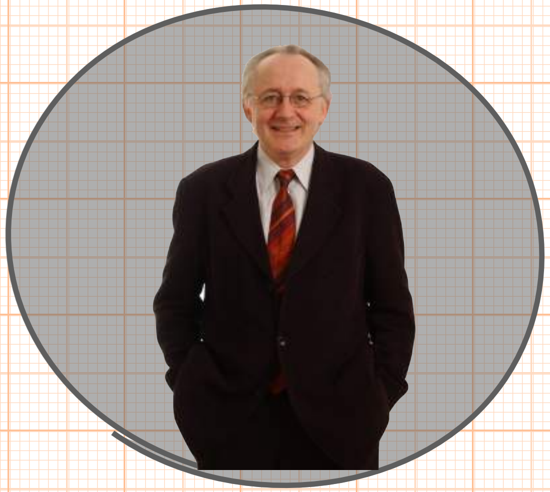
$$Col(prog) \subseteq P$$

- generally, the collecting semantics cannot be computed,
  we settle for a weaker property $S^\sharp$ that
  - is sound: $Col(prog) \subseteq S^\sharp$
  - implies the desired property: $S^\sharp \subseteq P$

Course 2 | Program Semantics and Properties | Antoine Miné | p. 24 / 98

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

Property “being Patrick Cousot”



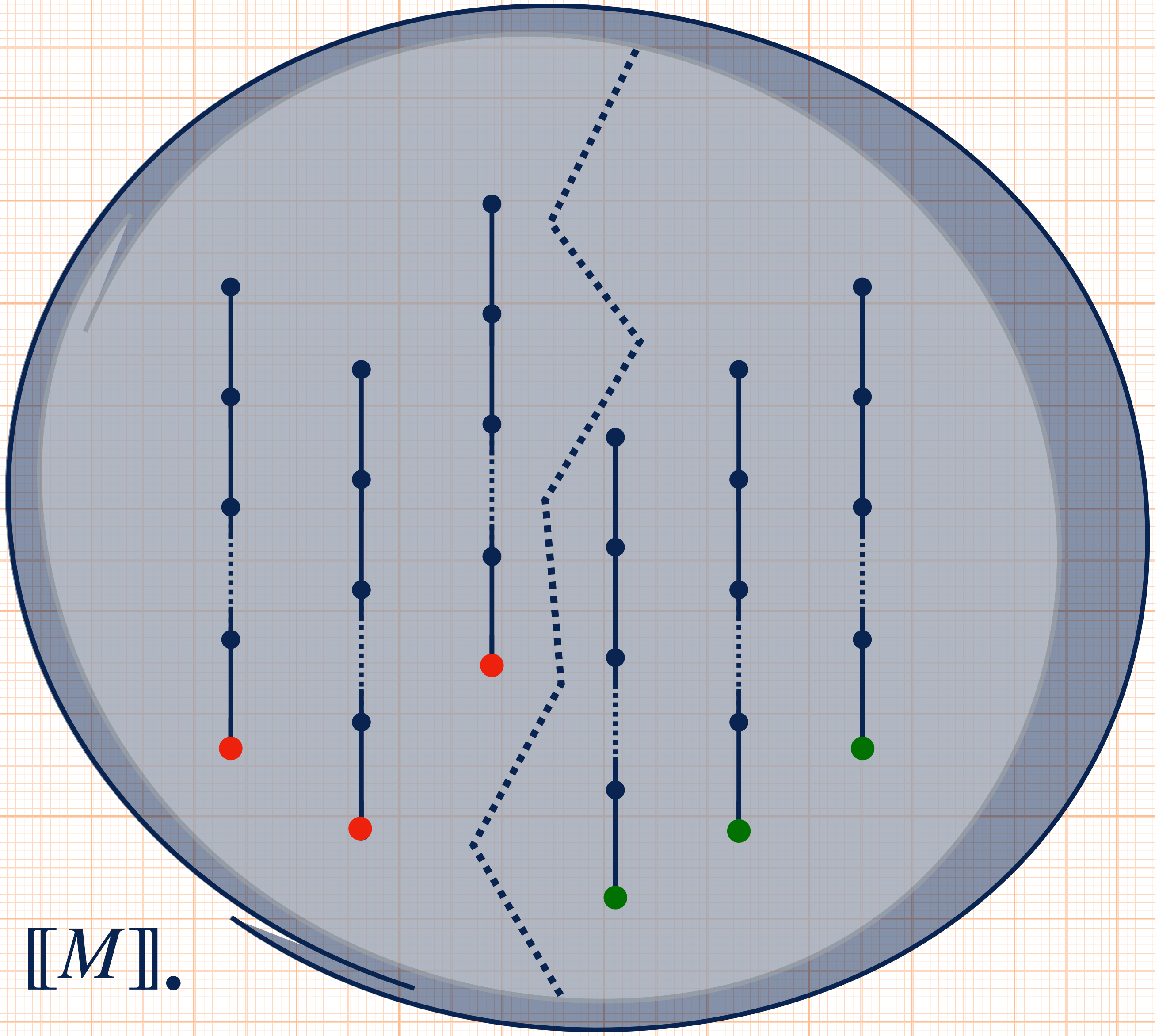
Property “being program P”

$$\{[\![P]\!]\}$$

(Another) Hierarchy of Semantics

$$\begin{tikzcd}
\llbracket M \rrbracket_{\rightsquigarrow} & \text{dependency semantics} \\
\llbracket M \rrbracket_{\bullet} \arrow[u, "\alpha_{\rightsquigarrow}"] & \text{outcome semantics} \\
\{\llbracket M \rrbracket\} \arrow[u, "\alpha_{\bullet}"] & \text{collecting semantics}
\end{tikzcd}$$

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

$$\mathbb{O} \overset{\text{def}}{=} \{\Sigma^{+}_{o_1=v_1,\dots,o_k=v_k} \mid v_1, \dots, v_k \in \mathscr{V}\} \cup \{\Sigma^{\omega}\}$$

outcomes

Lemma

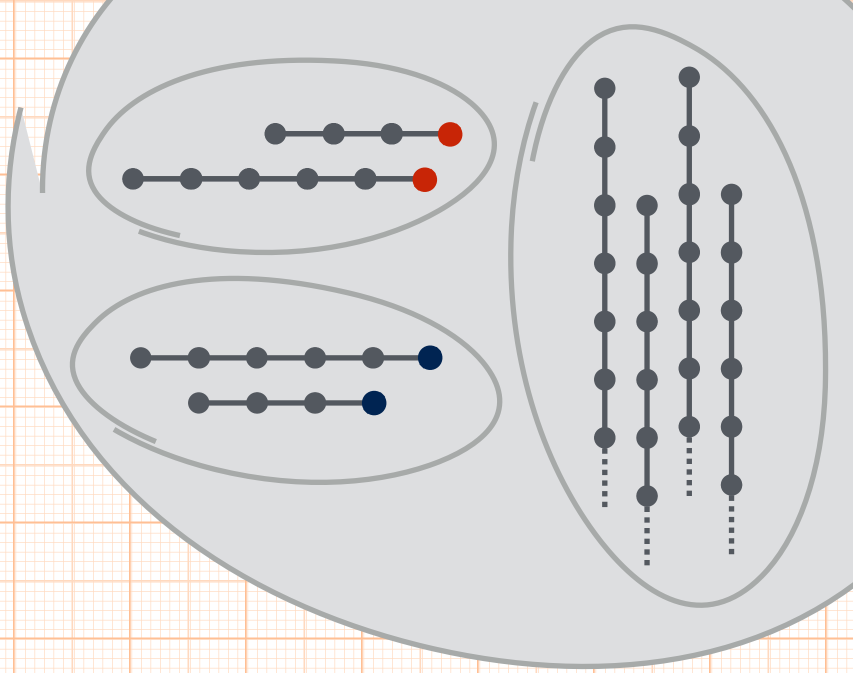
$$P \models \mathscr{N}_J \Leftrightarrow \{[\![P]\!] \cap O \mid O \in \mathbb{O}\} \subseteq \mathscr{N}_J$$

$$\begin{tikzcd}
\langle \mathscr{P}(\mathscr{P}(\Sigma^{+\infty})), \subseteq \rangle \arrow[r, bend right, "\alpha_{\bullet}"'] & \langle \mathscr{P}(\mathscr{P}(\Sigma^{+\infty})), \subseteq_{\bullet} \rangle \arrow[l, bend right, "\gamma_{\bullet}"']
\end{tikzcd}$$

$$\alpha_{\bullet}(S) \overset{\text{def}}{=} \{T \cap O \mid T \in S \wedge O \in \mathbb{O}\}$$

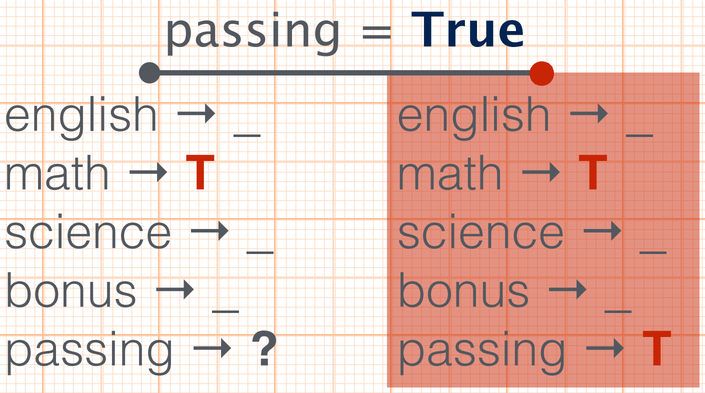
outcome abstraction

Outcome Semantics



$[\![P]\!]_{\bullet}$

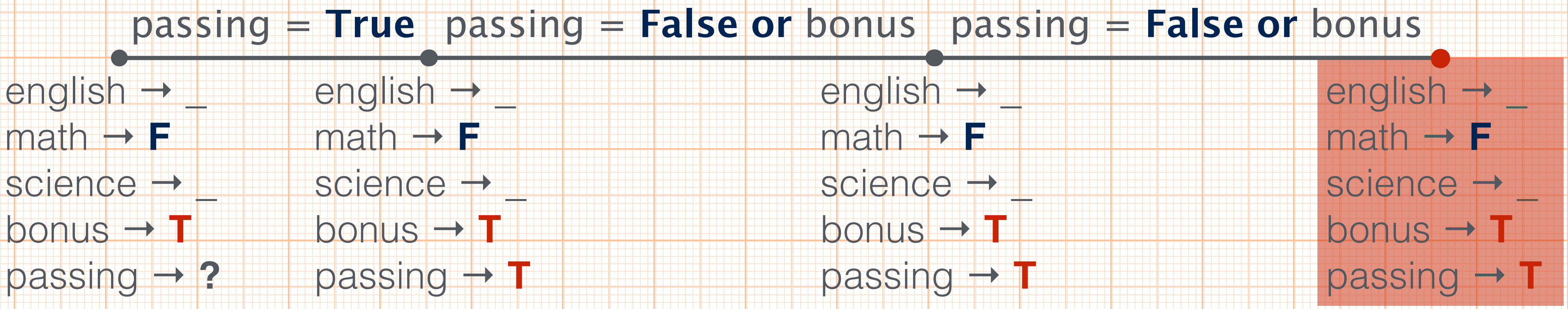
```
passing = True
if not english:
    english = False
if not math:
    passing = False or bonus
if not math:
    passing = False or bonus
```



passing = **True**

english → _
math → **T**
science → _
bonus → _
passing → **?**

english → _
math → **T**
science → _
bonus → _
passing → **T**



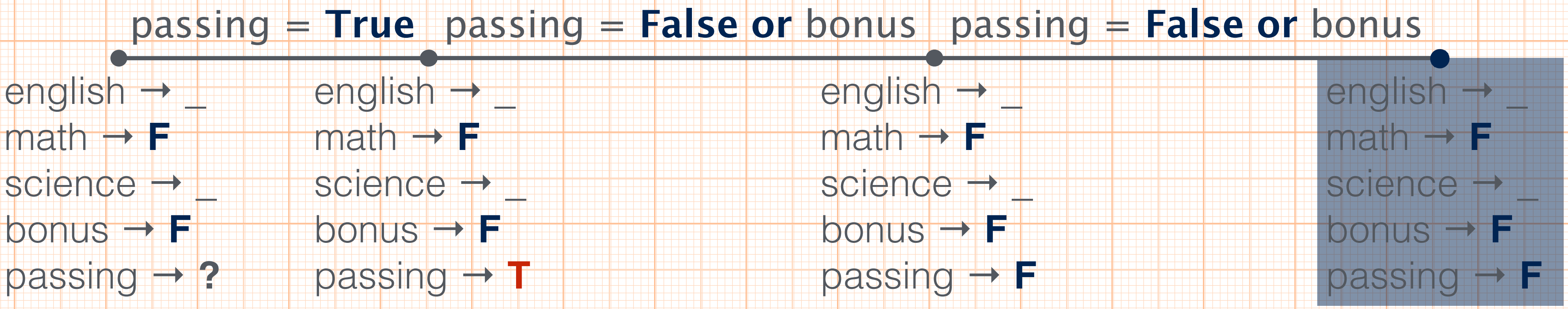
passing = **True** passing = **False or** bonus passing = **False or** bonus

english → _
math → **F**
science → _
bonus → **T**
passing → **?**

english → _
math → **F**
science → _
bonus → **T**
passing → **T**

english → _
math → **F**
science → _
bonus → **T**
passing → **T**

english → _
math → **F**
science → _
bonus → **T**
passing → **T**



passing = **True** passing = **False or** bonus passing = **False or** bonus

english → _
math → **F**
science → _
bonus → **F**
passing → **?**

english → _
math → **F**
science → _
bonus → **F**
passing → **T**

english → _
math → **F**
science → _
bonus → **F**
passing → **F**

english → _
math → **F**
science → _
bonus → **F**
passing → **F**

Outcome Semantics

$$\mathbb{O} \overset{\text{def}}{=} \{\Sigma^{+}_{o_1=v_1,\ldots,o_k=v_k} \mid v_1,\ldots,v_k \in \mathscr{V}\} \cup \{\Sigma^{\omega}\}$$ **outcomes**

Lemma

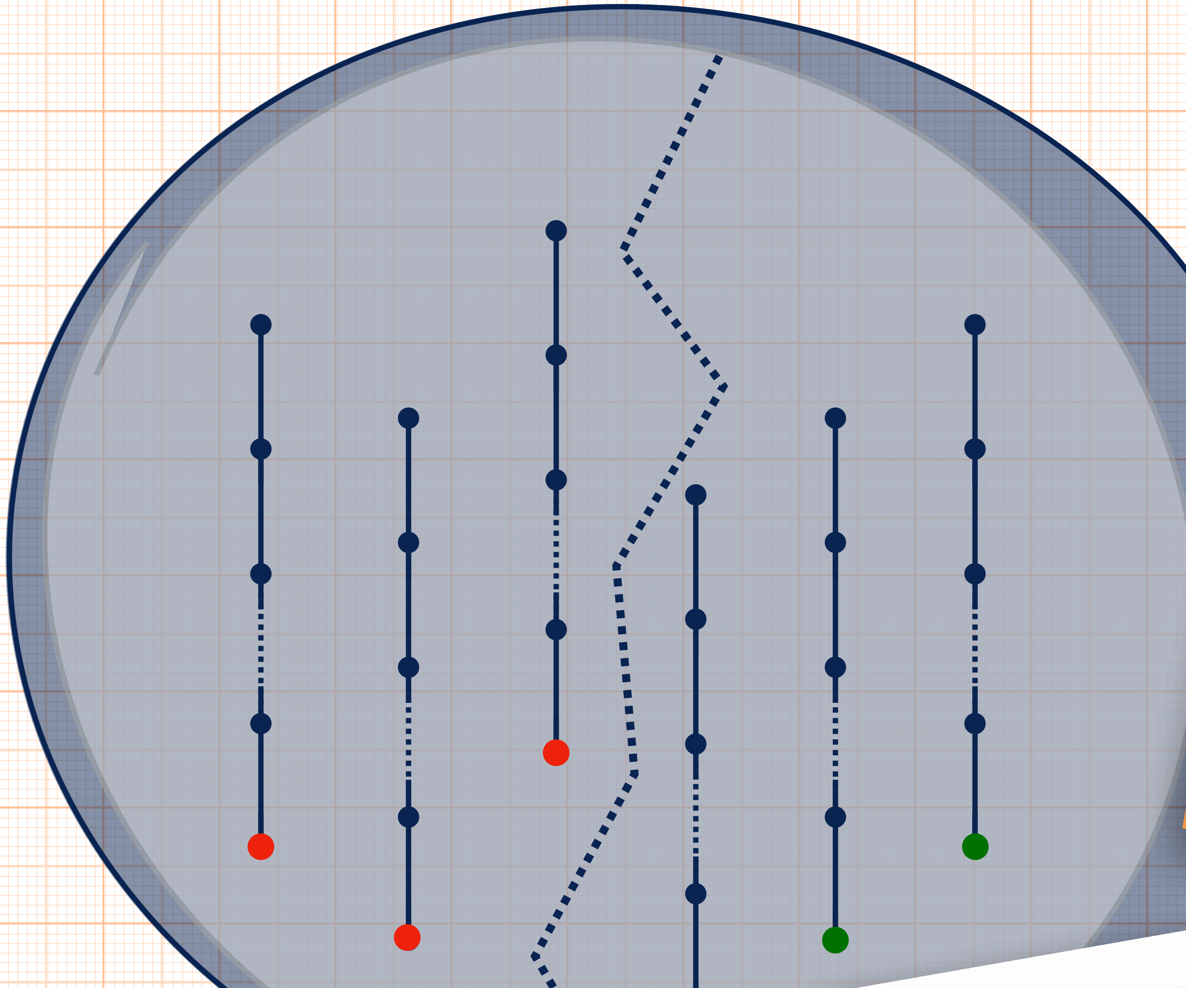
$$P \models \mathscr{N}_J \Leftrightarrow \{[\![P]\!] \cap O \mid O \in \mathbb{O}\} \subseteq \mathscr{N}_J$$

$$\begin{tikzcd}
\langle \mathscr{P}(\mathscr{P}(\Sigma^{+\infty})), \subseteq \rangle \arrow[r, bend right, "\alpha_\bullet"'] & \langle \mathscr{P}(\mathscr{P}(\Sigma^{+\infty})), \subseteq_\bullet \rangle \arrow[l, bend right, "\gamma_\bullet"']
\end{tikzcd}$$

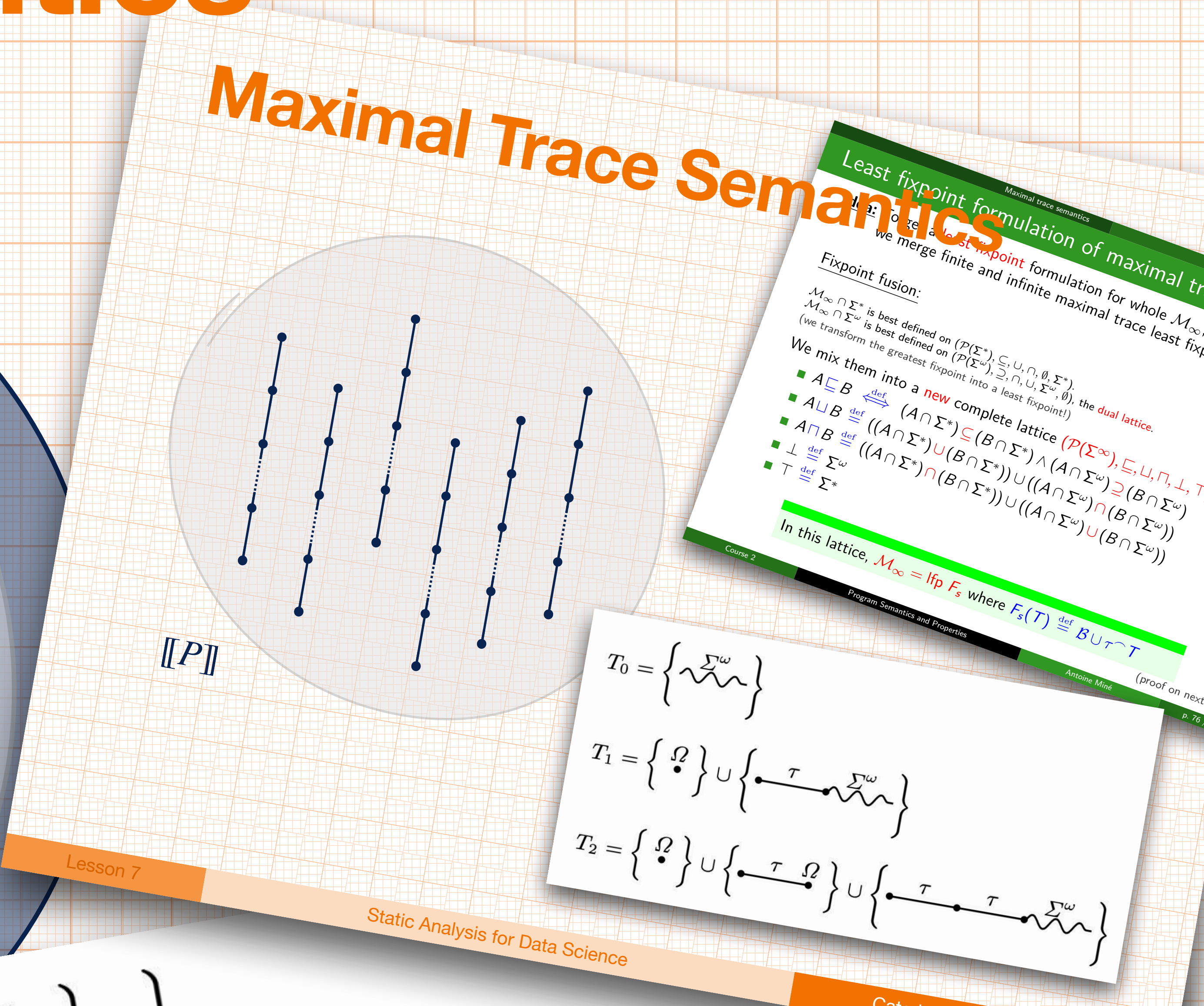
$$\alpha_\bullet(S) \overset{\text{def}}{=} \{T \cap O \mid T \in S \wedge O \in \mathbb{O}\}$$ **outcome abstraction**

$$[\![P]\!]_\bullet \overset{\text{def}}{=} \alpha_\bullet(\{[\![P]\!]\}) = \{[\![P]\!] \cap O \mid O \in \mathbb{O}\}$$

Outcome Semantics



$[\![P]\!]$.



Maximal Trace Semantics

$[\![P]\!]$

Maximal trace semantics

Least fixpoint formulation of maximal tr

Idea: To get a least fixpoint formulation for whole $\mathcal{M}_\infty$,
we merge finite and infinite maximal trace least fixp

Fixpoint fusion:

$\mathcal{M}_\infty \cap \Sigma^*$ is best defined on $(\mathcal{P}(\Sigma^*), \subseteq, \cup, \cap, \emptyset, \Sigma^*)$.
$\mathcal{M}_\infty \cap \Sigma^\omega$ is best defined on $(\mathcal{P}(\Sigma^\omega), \supseteq, \cap, \cup, \Sigma^\omega, \emptyset)$, the dual lattice.
(we transform the greatest fixpoint into a least fixpoint!)

We mix them into a new complete lattice $(\mathcal{P}(\Sigma^\infty), \sqsubseteq, \sqcup, \sqcap, \bot, \top$

- $A \sqsubseteq B \overset{\text{def}}{\iff} (A \cap \Sigma^*) \subseteq (B \cap \Sigma^*) \wedge (A \cap \Sigma^\omega) \supseteq (B \cap \Sigma^\omega)$
- $A \sqcup B \overset{\text{def}}{=} ((A \cap \Sigma^*) \cup (B \cap \Sigma^*)) \cup ((A \cap \Sigma^\omega) \cap (B \cap \Sigma^\omega))$
- $A \sqcap B \overset{\text{def}}{=} ((A \cap \Sigma^*) \cap (B \cap \Sigma^*)) \cup ((A \cap \Sigma^\omega) \cup (B \cap \Sigma^\omega))$
- $\bot \overset{\text{def}}{=} \Sigma^\omega$
- $\top \overset{\text{def}}{=} \Sigma^*$

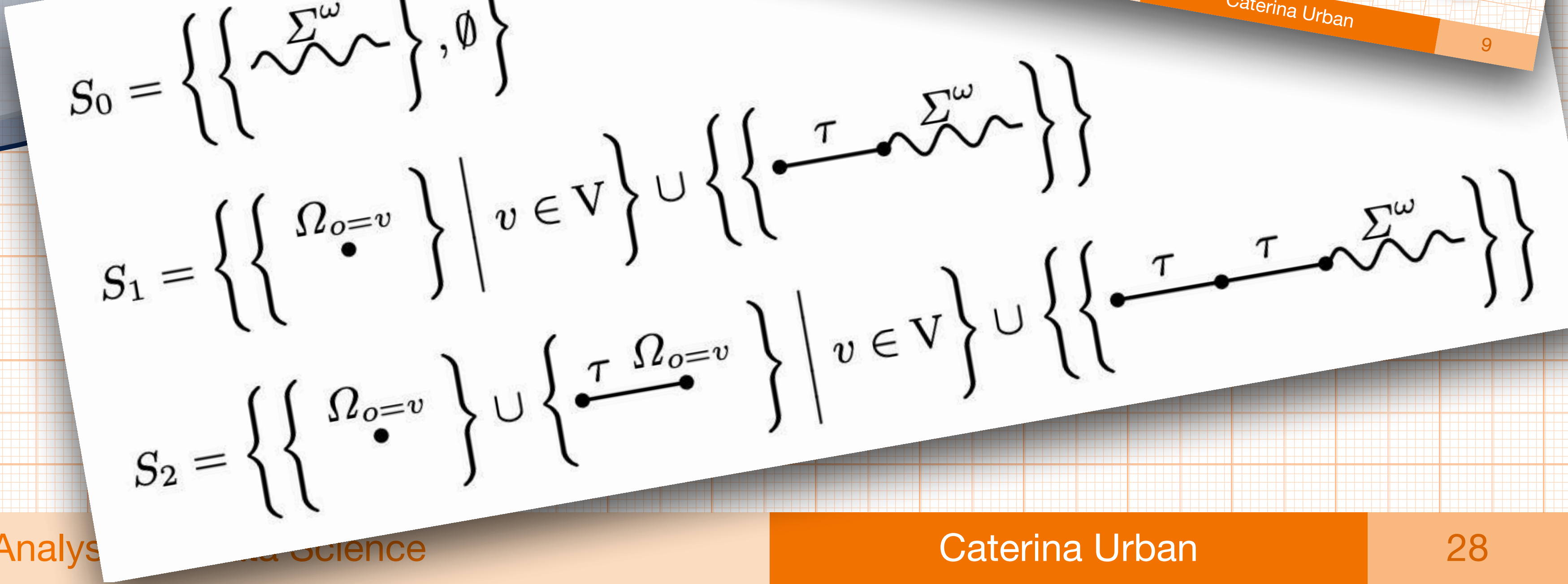
In this lattice, $\mathcal{M}_\infty = \text{lfp } F_s$ where $F_s(T) \overset{\text{def}}{=} \mathcal{B} \cup \tau \frown T$

(proof on next

$$T_0 = \{ \overset{\Sigma^\omega}{\rightsquigarrow} \}$$

$$T_1 = \{ \overset{\Omega}{\bullet} \} \cup \{ \bullet \xrightarrow{\tau} \bullet \overset{\Sigma^\omega}{\rightsquigarrow} \}$$

$$T_2 = \{ \overset{\Omega}{\bullet} \} \cup \{ \bullet \xrightarrow{\tau} \overset{\Omega}{\bullet} \} \cup \{ \bullet \xrightarrow{\tau} \bullet \xrightarrow{\tau} \bullet \overset{\Sigma^\omega}{\rightsquigarrow} \}$$



$$S_0 = \{ \{ \overset{\Sigma^\omega}{\rightsquigarrow} \}, \emptyset \}$$

$$S_1 = \{ \{ \overset{\Omega_{o=v}}{\bullet} \} \mid v \in \mathbb{V} \} \cup \{ \{ \bullet \xrightarrow{\tau} \bullet \overset{\Sigma^\omega}{\rightsquigarrow} \} \}$$

$$S_2 = \{ \{ \overset{\Omega_{o=v}}{\bullet} \} \cup \{ \bullet \xrightarrow{\tau} \overset{\Omega_{o=v}}{\bullet} \} \mid v \in \mathbb{V} \} \cup \{ \{ \bullet \xrightarrow{\tau} \bullet \xrightarrow{\tau} \bullet \overset{\Sigma^\omega}{\rightsquigarrow} \} \}$$

Outcome Semantics

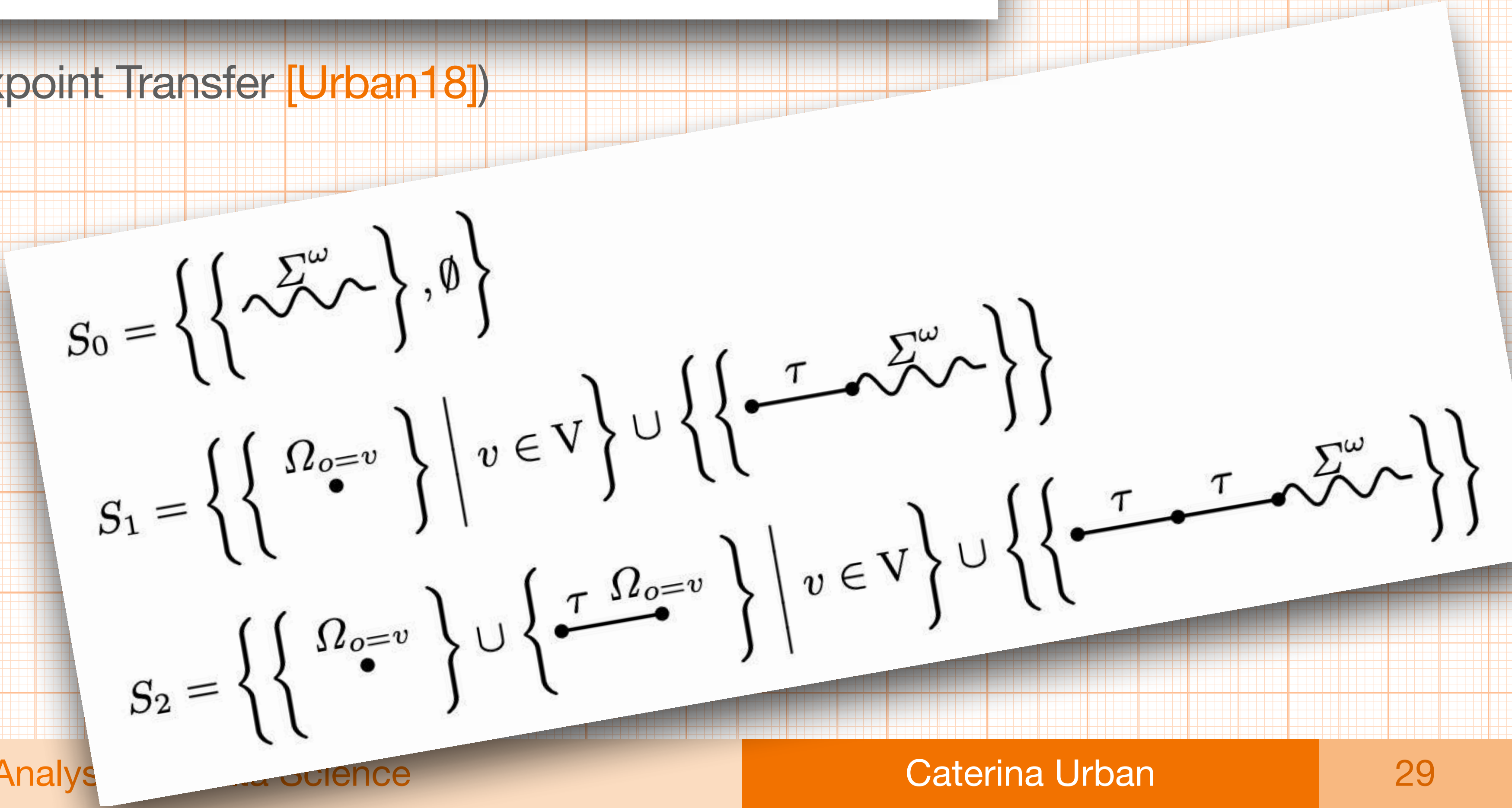
$$S_1 \mathrel{\dot{\sqsubseteq}} S_2 \overset{\text{def}}{=} \bigwedge_{v_1,\ldots,v_k \in \mathrm{V}} {S_1}^{+}_{o_1=v_1,\ldots,o_k=v_k} \subseteq {S_2}^{+}_{o_1=v_1,\ldots,o_k=v_k} \wedge S_1^\omega \supseteq S_2^\omega$$

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

$$\begin{aligned} \Lambda_\bullet &= \mathrm{lfp}^{\dot{\sqsubseteq}}\, \Theta_\bullet \\ \Theta_\bullet(S) &\overset{\text{def}}{=} \{\Omega_{o_1=v_1,\ldots,o_k=v_k} \mid v_1,\ldots,v_k \in \mathrm{V}\} \mathbin{\dot{\cup}} \{\tau ; T \mid T \in S\} \end{aligned} \tag{9}$$

where $S_1 \mathbin{\dot{\cup}} S_2 \overset{\text{def}}{=} \{{S_1}^{+}_{o_1=v_1,\ldots,o_k=v_k} \cup {S_2}^{+}_{o_1=v_1,\ldots,o_k=v_k} \mid v_1,\ldots,v_k \in \mathrm{V}\} \cup S_1^\omega \cup S_2^\omega$.

(proof by Kleenian Fixpoint Transfer [Urban18])

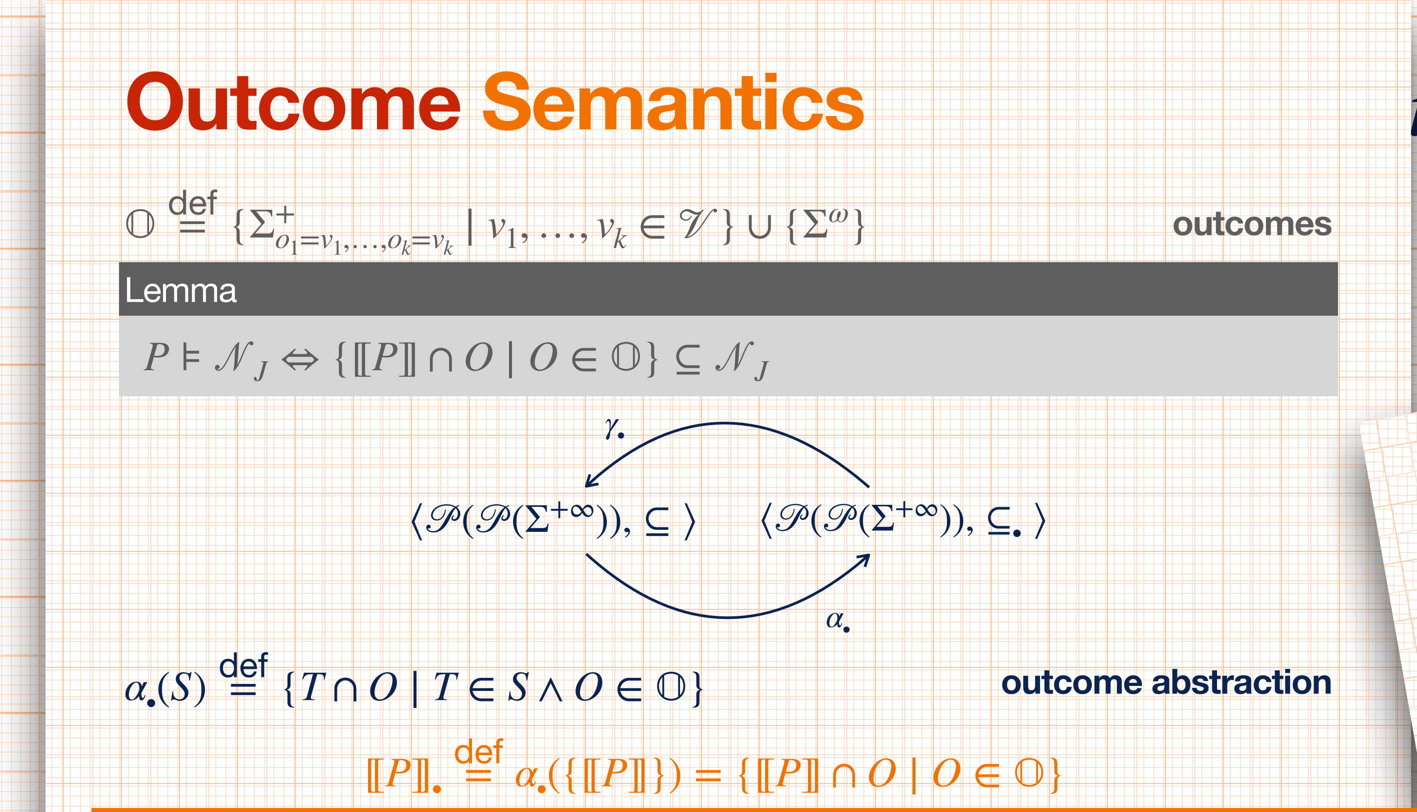


Input Data (Non-)Usage

$$\mathscr{N}_J \overset{\text{def}}{=} \{[\![P]\!] \in \mathscr{P}(\Sigma^{+\infty}) \mid \text{UNUSED}_J([\![P]\!])\}$$

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

$$t_0(J) \neq V \Rightarrow \exists t' \in [\![P]\!]:$$
$$J \Rightarrow t_0(i) = t'_0(i))$$



Outcome Semantics

$\mathbb{O} \overset{\text{def}}{=} \{\Sigma^+_{o_1=v_1,\ldots,o_k=v_k} \mid v_1, \ldots, v_k \in \mathscr{V}\} \cup \{\Sigma^\omega\}$ **outcomes**

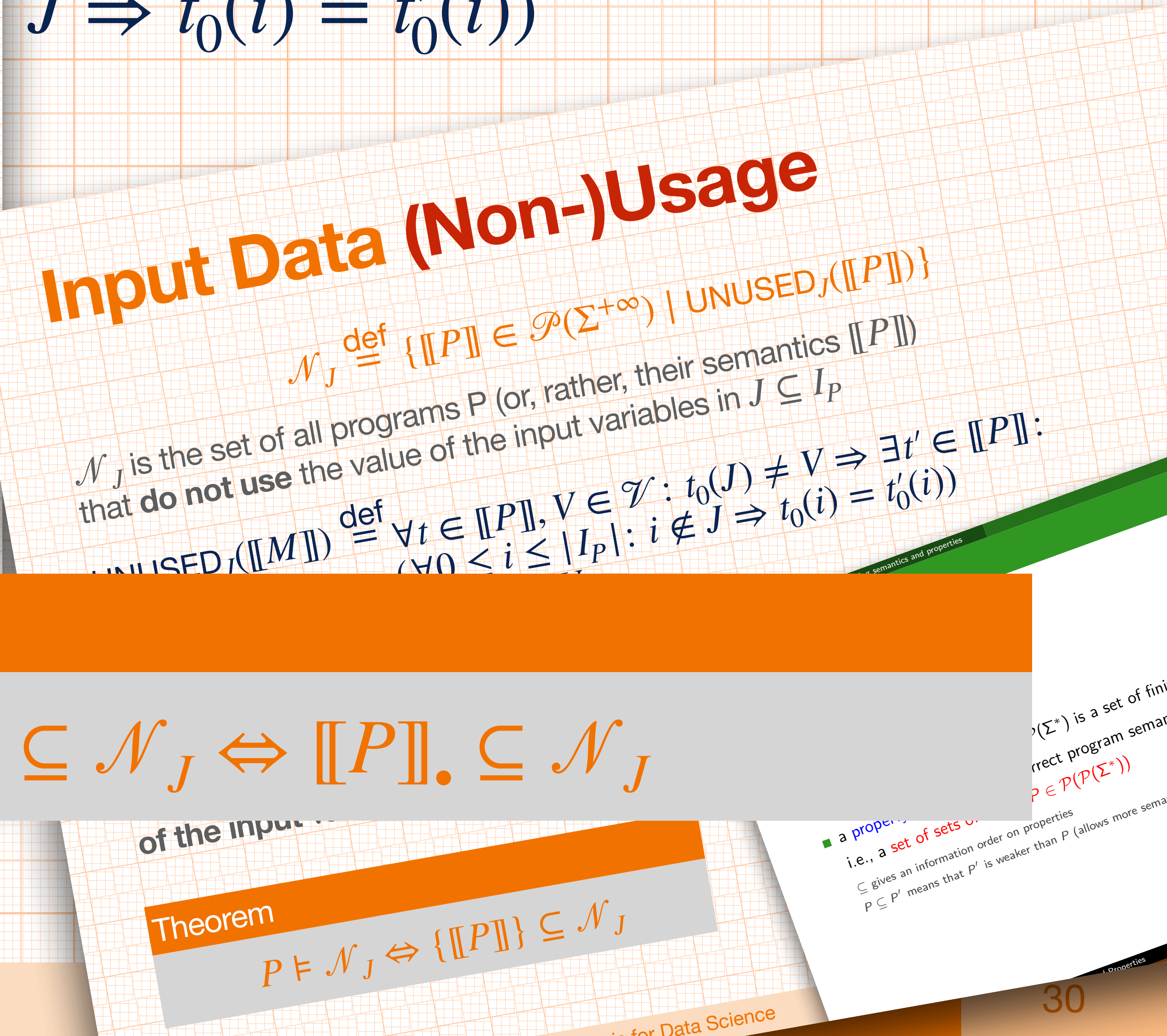
Lemma

$P \models \mathscr{N}_J \Leftrightarrow \{[\![P]\!] \cap O \mid O \in \mathbb{O}\} \subseteq \mathscr{N}_J$

$$\begin{tikzcd}
\langle \mathscr{P}(\mathscr{P}(\Sigma^{+\infty})), \subseteq \rangle \arrow[r, bend right, "\alpha_\bullet"'] & \langle \mathscr{P}(\mathscr{P}(\Sigma^{+\infty})), \subseteq_\bullet \rangle \arrow[l, bend right, "\gamma_\bullet"']
\end{tikzcd}$$

$\alpha_\bullet(S) \overset{\text{def}}{=} \{T \cap O \mid T \in S \wedge O \in \mathbb{O}\}$ **outcome abstraction**

$$[\![P]\!]_\bullet \overset{\text{def}}{=} \alpha_\bullet(\{[\![P]\!]\}) = \{[\![P]\!] \cap O \mid O \in \mathbb{O}\}$$



Input Data (Non-)Usage

$\mathscr{N}_J \overset{\text{def}}{=} \{[\![P]\!] \in \mathscr{P}(\Sigma^{+\infty}) \mid \text{UNUSED}_J([\![P]\!])\}$

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

$\text{UNUSED}_J([\![M]\!]) \overset{\text{def}}{=} \forall t \in [\![P]\!], V \in \mathscr{V}: t_0(J) \neq V \Rightarrow \exists t' \in [\![P]\!]:$
$(\forall 0 < i \leq |I_P|: i \notin J \Rightarrow t_0(i) = t'_0(i))$

of the input

Theorem

$P \models \mathscr{N}_J \Leftrightarrow \{[\![P]\!]\} \subseteq \mathscr{N}_J$

for Data Science

semantics and properties

$\mathcal{P}(\Sigma^*)$ is a set of fini

rrect program seman

$P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$

- a proper
  i.e., a set of sets o
  $\subseteq$ gives an information order on properties
  $P \subseteq P'$ means that P' is weaker than P (allows more sema

Properties

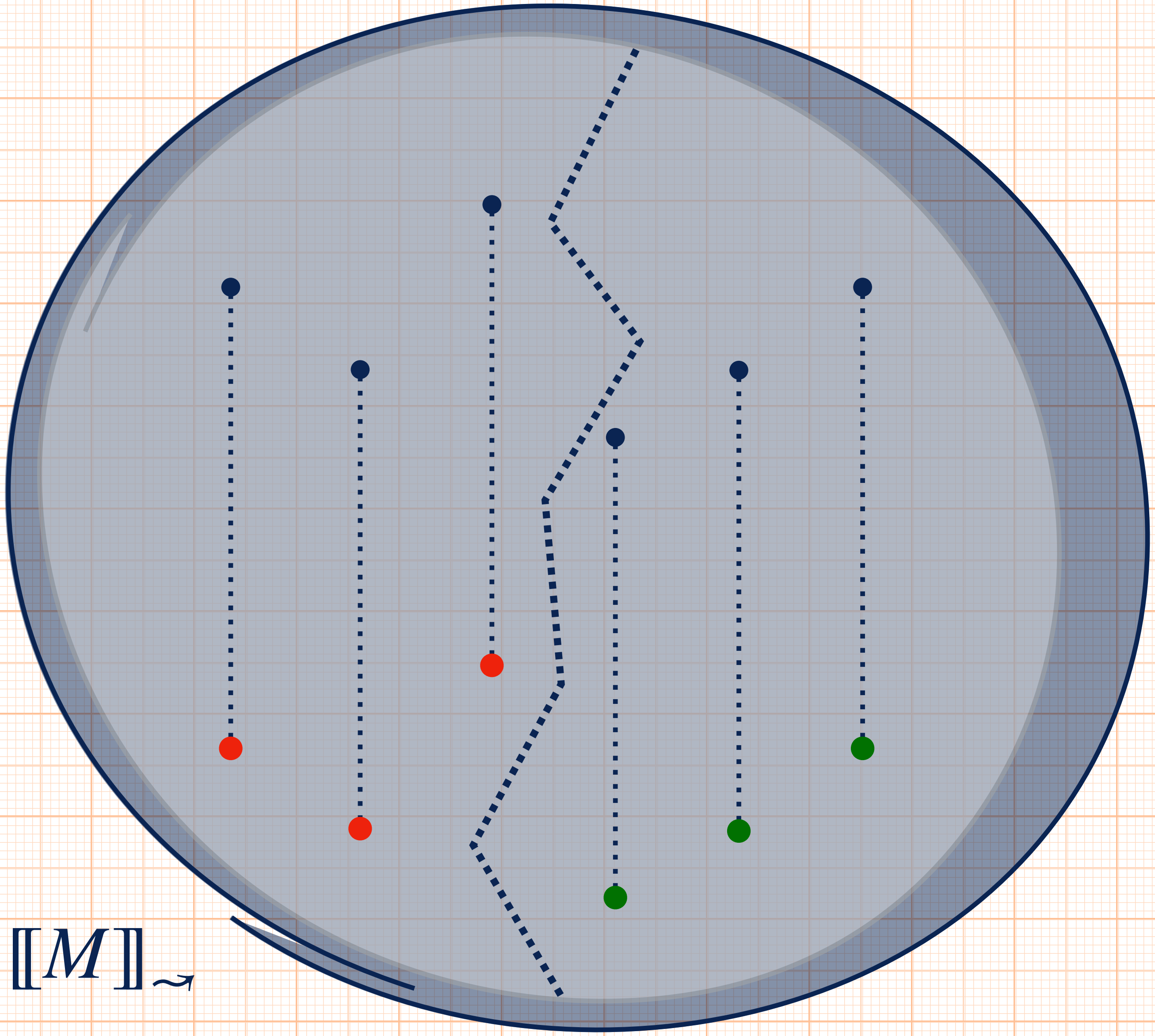
Theorem

$$P \models \mathscr{N}_J \Leftrightarrow \{[\![P]\!]\} \subseteq \mathscr{N}_J \Leftrightarrow \alpha_\bullet(\{[\![P]\!]\}) \subseteq \mathscr{N}_J \Leftrightarrow [\![P]\!]_\bullet \subseteq \mathscr{N}_J$$

(Another) Hierarchy of Semantics

$$\begin{tikzcd}
\llbracket M \rrbracket_{\rightsquigarrow} & \text{dependency semantics} \\
\llbracket M \rrbracket_{\bullet} \arrow[u, "\alpha_{\rightsquigarrow}"'] & \text{outcome semantics} \\
\{ \llbracket M \rrbracket \} \arrow[u, "\alpha_{\bullet}"'] & \text{collecting semantics}
\end{tikzcd}$$

Dependency Semantics



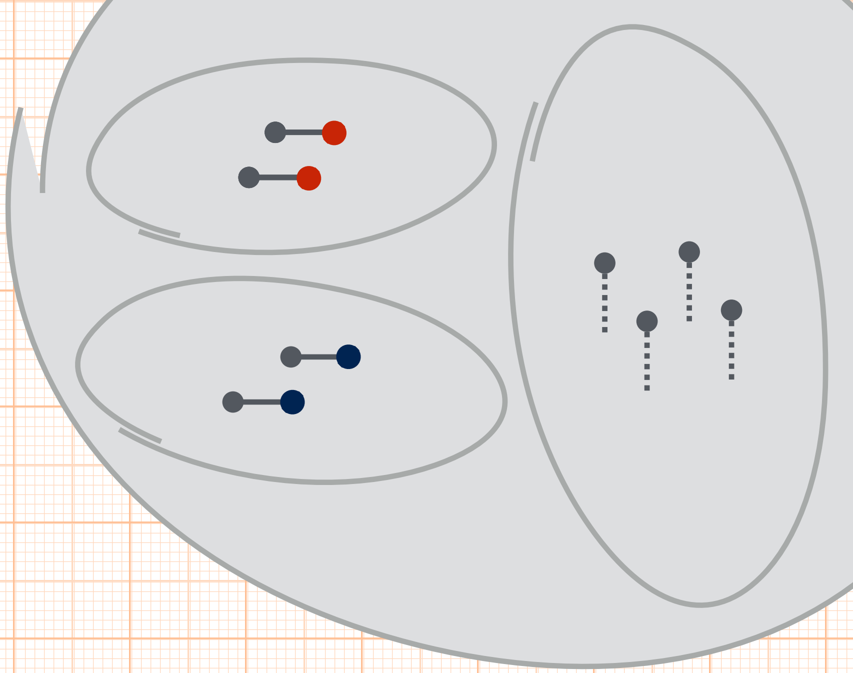
💡 to reason about input data (non-)usage **we do not need to consider all intermediate computations** between the initial and final states of a trace (if any)

Dependency Semantics

$$\begin{tikzcd}
\langle \mathscr{P}(\mathscr{P}(\Sigma^{+\infty})), \subseteq_{\bullet} \rangle \arrow[r, bend right, "\alpha_{\rightsquigarrow}"'] & \langle \mathscr{P}(\mathscr{P}(\Sigma \times \Sigma_{\perp})), \subseteq_{\rightsquigarrow} \rangle \arrow[l, bend right, "\gamma_{\rightsquigarrow}"']
\end{tikzcd}$$

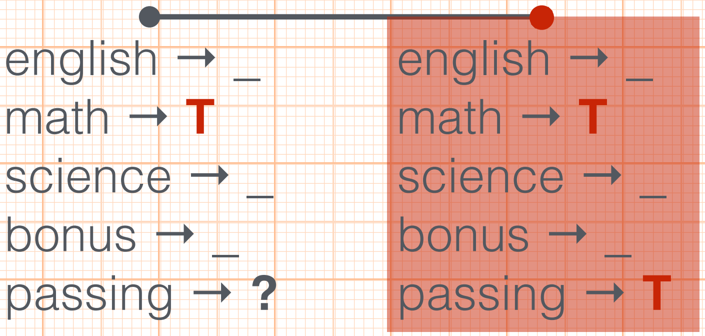
$$\alpha_{\rightsquigarrow}(S) \overset{\text{def}}{=} \{ \{ \langle t_0, t_\omega \rangle \in \Sigma \times \Sigma_{\perp} \mid t \in T \} \mid T \in S \}$$

Dependency Semantics



$[\![P]\!]_{\rightsquigarrow}$

```
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 → ?

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

english → _
math → F
science → _
bonus → T
passing → ?

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

english → _
math → F
science → _
bonus → F
passing → ?

english → _
math → F
science → _
bonus → F
passing → F

Dependency Semantics

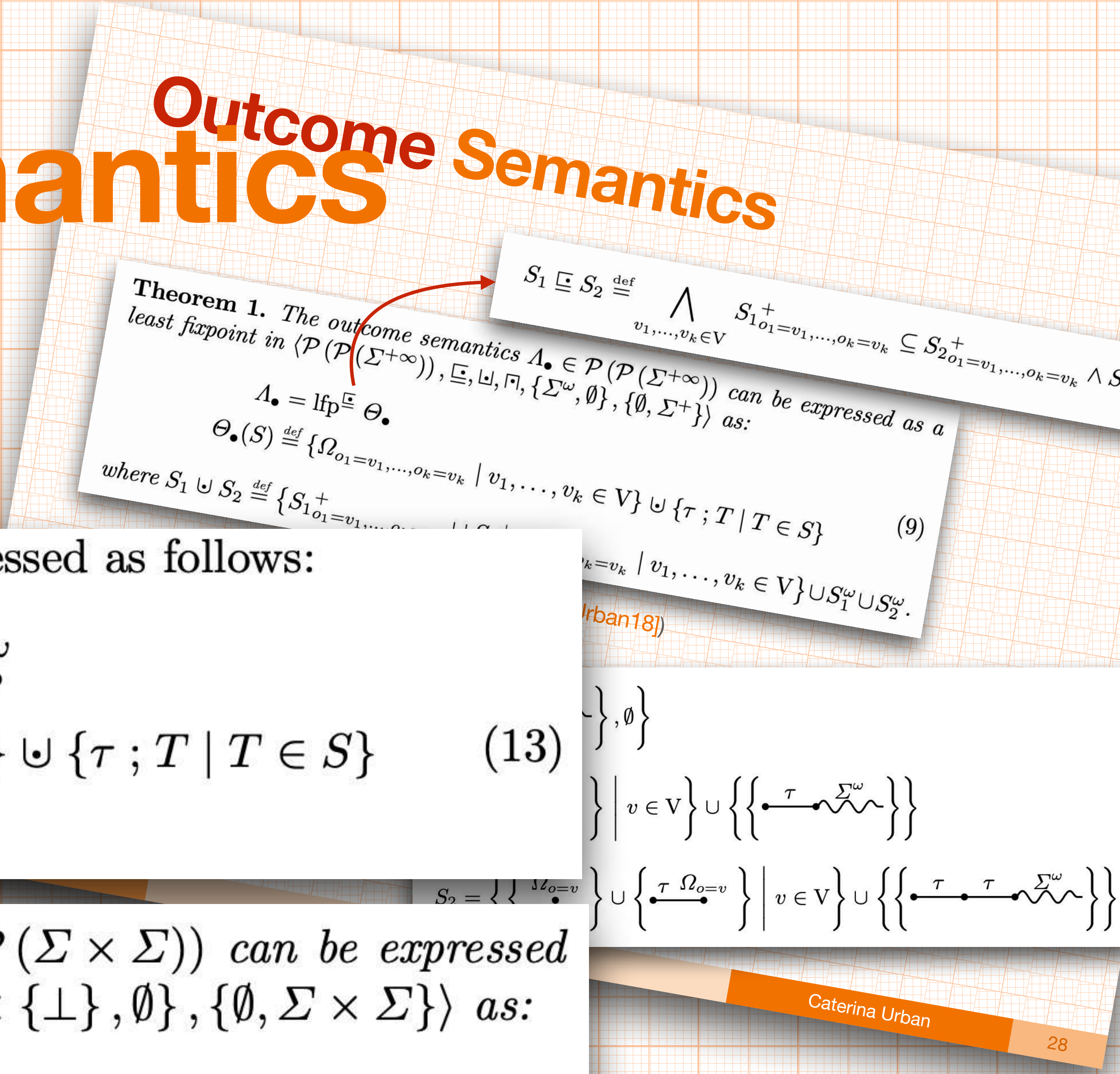
$$\begin{tikzcd}
\langle \mathscr{P}(\mathscr{P}(\Sigma^{+\infty})), \subseteq_{\bullet} \rangle \arrow[r, bend right, "\alpha_{\rightsquigarrow}"'] & \langle \mathscr{P}(\mathscr{P}(\Sigma \times \Sigma_{\perp})), \subseteq_{\rightsquigarrow} \rangle \arrow[l, bend right, "\gamma_{\rightsquigarrow}"']
\end{tikzcd}$$

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

$$\gamma_{\rightsquigarrow}(S) \overset{\text{def}}{=} \{ T \in \mathscr{P}(\Sigma^{+\infty}) \mid \{ \langle t_0, t_\omega \rangle \in \Sigma \times \Sigma_{\perp} \mid t \in T \} \in S \}$$

$$[\![P]\!]_{\rightsquigarrow} \overset{\text{def}}{=} \alpha_{\rightsquigarrow}([\![P]\!]_{\bullet}) = \{ \{ \langle t_0, t_\omega \rangle \in \Sigma \times \Sigma \mid t \in [\![P]\!] \cap O \} \mid O \in \mathbb{O} \}$$

Dependency Semantics



Outcome Semantics

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

$$\Lambda_\bullet = \mathrm{lfp}^{\dot\sqsubseteq}\, \Theta_\bullet$$

$$\Theta_\bullet(S) \overset{\text{def}}{=} \{\Omega_{o_1=v_1,\ldots,o_k=v_k} \mid v_1,\ldots,v_k \in \mathrm{V}\} \mathrel{\dot\cup} \{\tau \mathbin{;} T \mid T \in S\} \qquad (9)$$

where $S_1 \mathrel{\dot\cup} S_2 \overset{\text{def}}{=} \{S_{1\,o_1=v_1,\ldots}^{+}$

$_{k=v_k} \mid v_1,\ldots,v_k \in \mathrm{V}\} \cup S_1^\omega \cup S_2^\omega.$

$$S_1 \mathrel{\dot\sqsubseteq} S_2 \overset{\text{def}}{=} \bigwedge_{v_1,\ldots,v_k \in \mathrm{V}} S_{1\,o_1=v_1,\ldots,o_k=v_k}^{+} \subseteq S_{2\,o_1=v_1,\ldots,o_k=v_k}^{+} \wedge S$$

rban18])

$S_2 =$

Caterina Urban

28

the outcome semantics $\Lambda_\bullet$ can be equivalently expressed as follows:

$$\Lambda_\bullet = \Lambda_\bullet^+ \cup \Lambda_\bullet^\omega = \mathrm{lfp}_{\emptyset}^{\dot\sqsubseteq}\, \Theta_\bullet^+ \cup \mathrm{lfp}_{\{\Sigma^\omega\}}^{\dot\sqsubseteq}\, \Theta_\bullet^\omega$$

$$\Theta_\bullet^+(S) \overset{\text{def}}{=} \{\Omega_{o_1=v_1,\ldots,o_k=v_k} \mid v_1,\ldots,v_k \in \mathrm{V}\} \mathrel{\dot\cup} \{\tau \mathbin{;} T \mid T \in S\} \qquad (13)$$

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

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

$$\Lambda_{\rightsquigarrow}^+ = \mathrm{lfp}_{\{\emptyset\}}^{\dot\sqsubseteq}\, \Theta_{\rightsquigarrow}^+$$

$$\Theta_{\rightsquigarrow}^+(S) \overset{\text{def}}{=} \{\Omega_{o_1=v_1,\ldots,o_k=v_k} \times \Omega_{o_1=v_1,\ldots,o_k=v_k} \mid v_1,\ldots,v_k \in \mathrm{V}\} \mathrel{\dot\cup} \{\tau \circ R \mid R \in S\}$$

Proof (Sketch). By

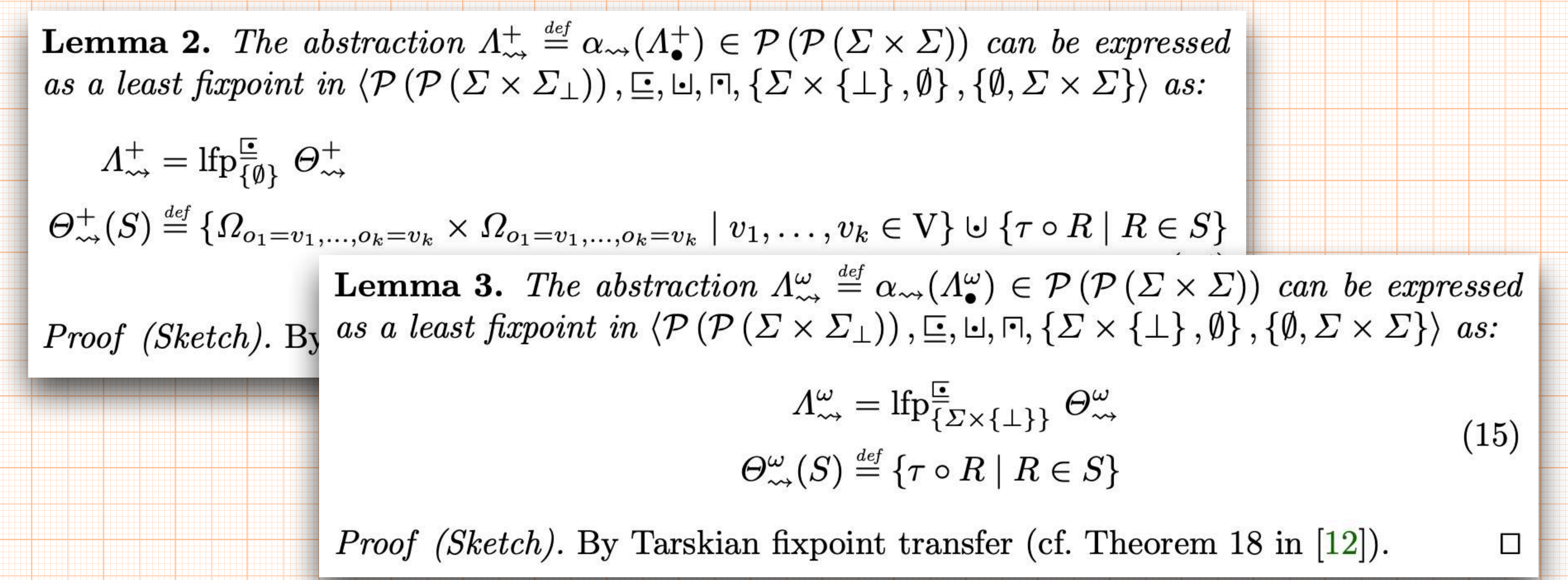
Lemma 3. *The abstraction* $\Lambda_{\rightsquigarrow}^\omega \overset{\text{def}}{=} \alpha_{\rightsquigarrow}(\Lambda_\bullet^\omega) \in \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma))$ *can be expressed as a least fixpoint in* $\langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma_\bot)), \mathrel{\dot\sqsubseteq}, \mathrel{\dot\sqcup}, \mathrel{\dot\sqcap}, \{\Sigma \times \{\bot\}, \emptyset\}, \{\emptyset, \Sigma \times \Sigma\}\rangle$ *as:*

$$\Lambda_{\rightsquigarrow}^\omega = \mathrm{lfp}_{\{\Sigma \times \{\bot\}\}}^{\dot\sqsubseteq}\, \Theta_{\rightsquigarrow}^\omega$$

$$\Theta_{\rightsquigarrow}^\omega(S) \overset{\text{def}}{=} \{\tau \circ R \mid R \in S\} \qquad (15)$$

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

Dependency Semantics



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

$$\Lambda^{+}_{\leadsto} = \mathrm{lfp}^{\dot{\sqsubseteq}}_{\{\emptyset\}}\ \Theta^{+}_{\leadsto}$$

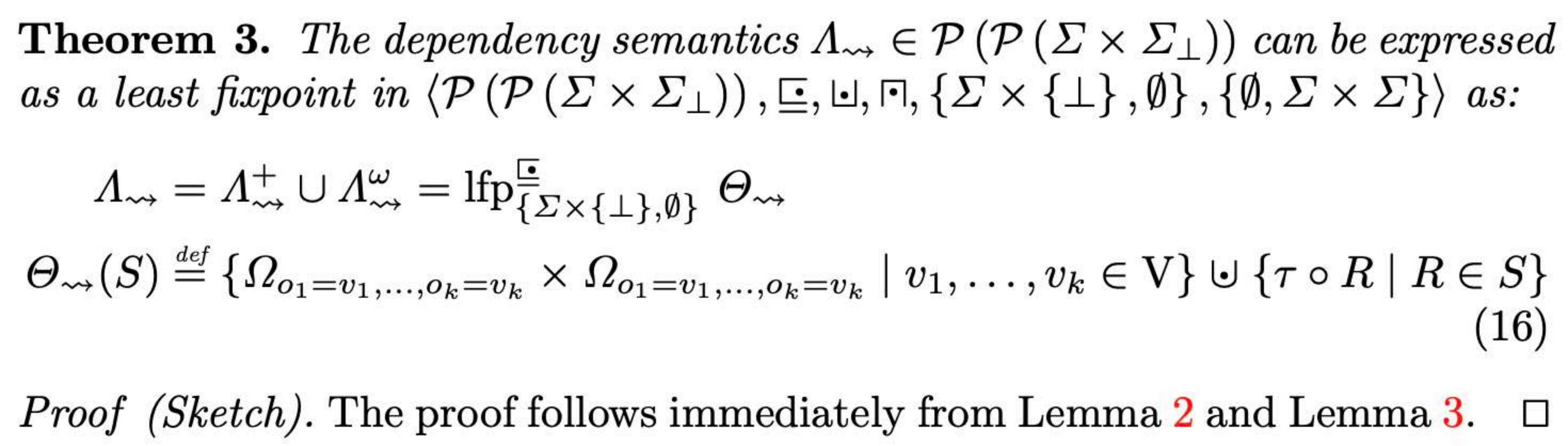
$$\Theta^{+}_{\leadsto}(S) \overset{def}{=} \{\Omega_{o_1=v_1,\dots,o_k=v_k} \times \Omega_{o_1=v_1,\dots,o_k=v_k} \mid v_1,\dots,v_k \in \mathrm{V}\} \mathbin{\dot{\cup}} \{\tau \circ R \mid R \in S\}$$

Proof (Sketch). By

Lemma 3. *The abstraction* $\Lambda^{\omega}_{\leadsto} \overset{def}{=} \alpha_{\leadsto}(\Lambda^{\omega}_{\bullet}) \in \mathcal{P}\left(\mathcal{P}\left(\Sigma \times \Sigma\right)\right)$ *can be expressed as a least fixpoint in* $\langle \mathcal{P}\left(\mathcal{P}\left(\Sigma \times \Sigma_{\perp}\right)\right), \dot{\sqsubseteq}, \dot{\sqcup}, \dot{\sqcap}, \{\Sigma \times \{\perp\}, \emptyset\}, \{\emptyset, \Sigma \times \Sigma\}\rangle$ *as:*

$$\begin{aligned} \Lambda^{\omega}_{\leadsto} &= \mathrm{lfp}^{\dot{\sqsubseteq}}_{\{\Sigma \times \{\perp\}\}}\ \Theta^{\omega}_{\leadsto} \\ \Theta^{\omega}_{\leadsto}(S) &\overset{def}{=} \{\tau \circ R \mid R \in S\} \end{aligned} \tag{15}$$

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



Theorem 3. *The dependency semantics* $\Lambda_{\leadsto} \in \mathcal{P}\left(\mathcal{P}\left(\Sigma \times \Sigma_{\perp}\right)\right)$ *can be expressed as a least fixpoint in* $\langle \mathcal{P}\left(\mathcal{P}\left(\Sigma \times \Sigma_{\perp}\right)\right), \dot{\sqsubseteq}, \dot{\sqcup}, \dot{\sqcap}, \{\Sigma \times \{\perp\}, \emptyset\}, \{\emptyset, \Sigma \times \Sigma\}\rangle$ *as:*

$$\Lambda_{\leadsto} = \Lambda^{+}_{\leadsto} \cup \Lambda^{\omega}_{\leadsto} = \mathrm{lfp}^{\dot{\sqsubseteq}}_{\{\Sigma \times \{\perp\}, \emptyset\}}\ \Theta_{\leadsto}$$

$$\Theta_{\leadsto}(S) \overset{def}{=} \{\Omega_{o_1=v_1,\dots,o_k=v_k} \times \Omega_{o_1=v_1,\dots,o_k=v_k} \mid v_1,\dots,v_k \in \mathrm{V}\} \mathbin{\dot{\cup}} \{\tau \circ R \mid R \in S\} \tag{16}$$

Proof (Sketch). The proof follows immediately from Lemma 2 and Lemma 3. ☐

Input Data (Non-)Usage

$$\mathcal{N}_J \overset{\text{def}}{=} \{[\![P]\!] \in \mathcal{P}(\Sigma^{+\infty}) \mid \text{UNUSED}_J([\![P]\!])\}$$

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

$$t_0(J) \neq V \Rightarrow \exists t' \in [\![P]\!] :$$
$$J \Rightarrow t_0(i) = t_0'(i))$$

Dependency Semantics

$$\begin{tikzcd}
\langle \mathcal{P}(\mathcal{P}(\Sigma^{+\infty})), \subseteq_{\bullet} \rangle \arrow[r, bend right, "\alpha_{\rightsquigarrow}"'] & \langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma_\perp)), \subseteq_{\rightsquigarrow} \rangle \arrow[l, bend right, "\gamma_{\rightsquigarrow}"']
\end{tikzcd}$$

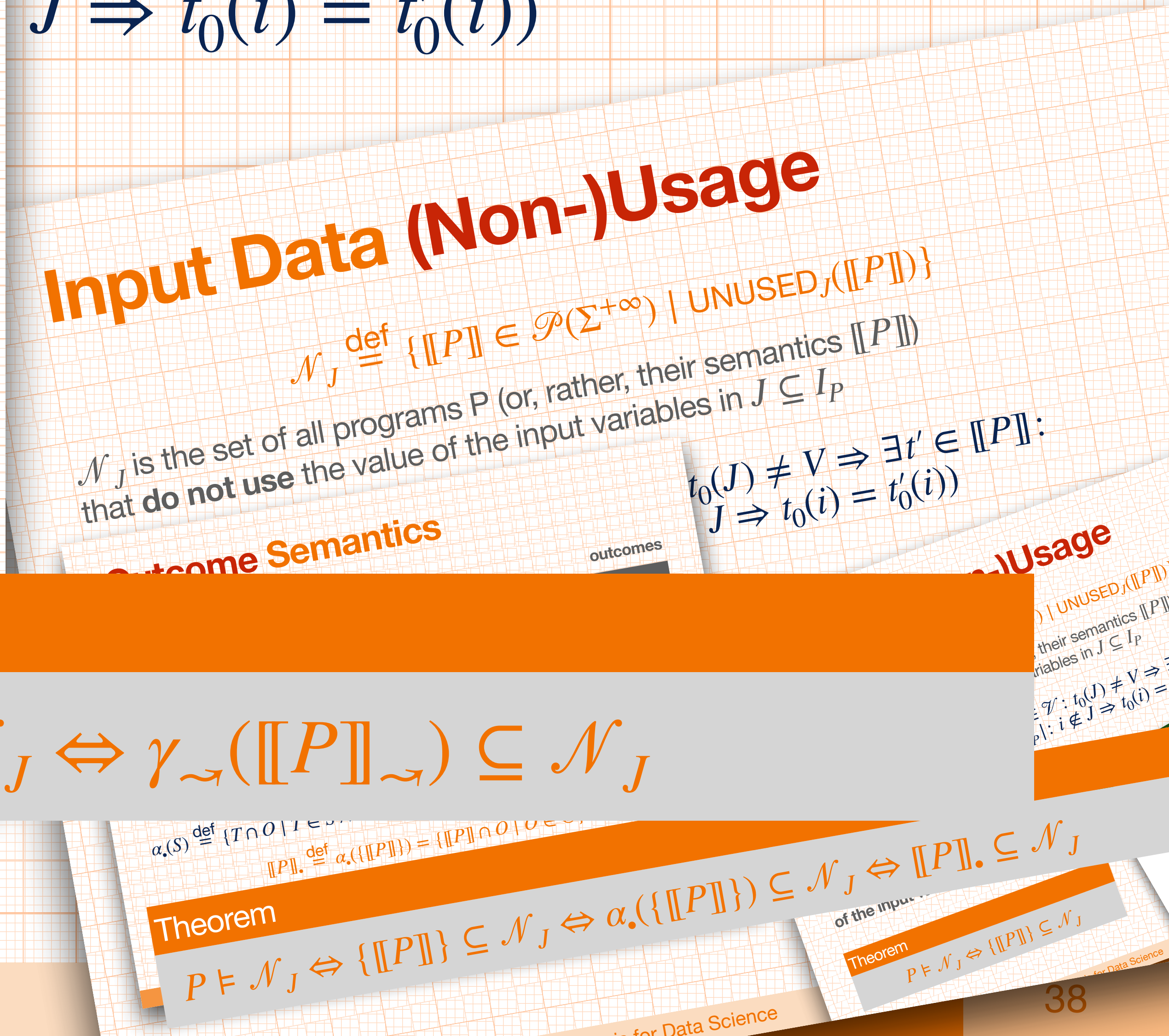
$$\alpha_{\rightsquigarrow}(S) \overset{\text{def}}{=} \{\{\langle t_0, t_\omega \rangle \in \Sigma \times \Sigma_\perp \mid t \in T\} \mid T \in S\}$$

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

$$[\![P]\!]_{\rightsquigarrow} \overset{\text{def}}{=} \alpha_{\rightsquigarrow}([\![P]\!]_{\bullet}) = \{\{\langle t_0, t_\omega \rangle \in \Sigma \times \Sigma \mid t \in [\![P]\!] \cap O\} \mid O \in \mathbb{O}\}$$

Theorem

$$P \models \mathcal{N}_J \Leftrightarrow \{[\![P]\!]\} \subseteq \mathcal{N}_J \Leftrightarrow [\![P]\!]_{\bullet} \subseteq \mathcal{N}_J \Leftrightarrow \gamma_{\rightsquigarrow}([\![P]\!]_{\rightsquigarrow}) \subseteq \mathcal{N}_J$$



Input Data (Non-)Usage

$\mathcal{N}_J \overset{\text{def}}{=} \{[\![P]\!] \in \mathcal{P}(\Sigma^{+\infty}) \mid \text{UNUSED}_J([\![P]\!])\}$

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

$t_0(J) \neq V \Rightarrow \exists t' \in [\![P]\!]:$
$J \Rightarrow t_0(i) = t_0'(i))$

Outcome Semantics

outcomes

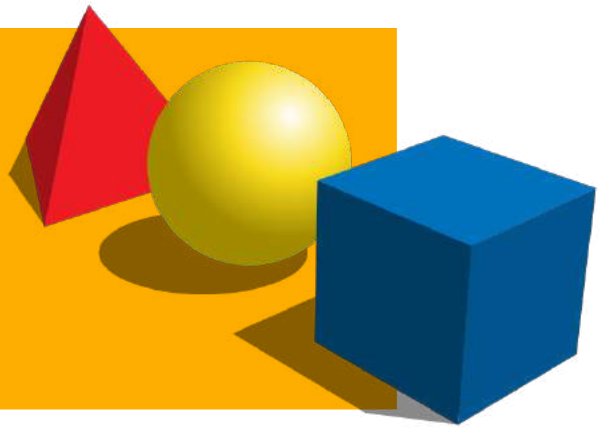
Theorem

$P \models \mathcal{N}_J \Leftrightarrow \{[\![P]\!]\} \subseteq \mathcal{N}_J \Leftrightarrow \alpha_{\bullet}(\{[\![P]\!]\}) \subseteq \mathcal{N}_J \Leftrightarrow [\![P]\!]_{\bullet} \subseteq \mathcal{N}_J$

Unused Data Analysis

practical tools
targeting specific programs

algorithmic approaches
to decide program properties



mathematical models
of the program behavior

Input Data (Non-)Usage Abstractions

Over-Approximation of the Used Input Data

⇒ Under-Approximation of the Unused Input Data

$$P \models \mathcal{N}_{J^\natural \subseteq J} \Leftarrow \gamma_{\rightsquigarrow}(\gamma_A(\llbracket P \rrbracket_A)) \subseteq \mathcal{N}_{J^\natural \subseteq J}$$

Secure Information Flow

L ⇢ x

L ⇢ y

H ⇢ t

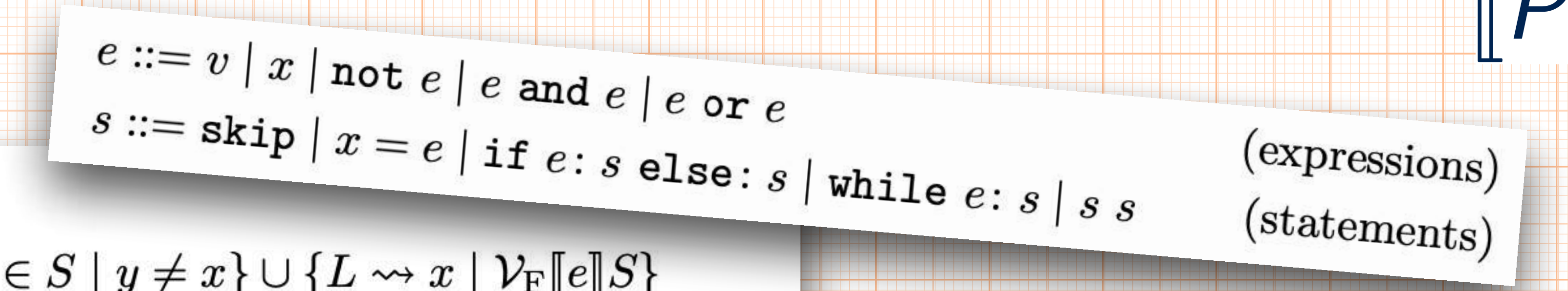
L ⇢ z

H ⇢ w

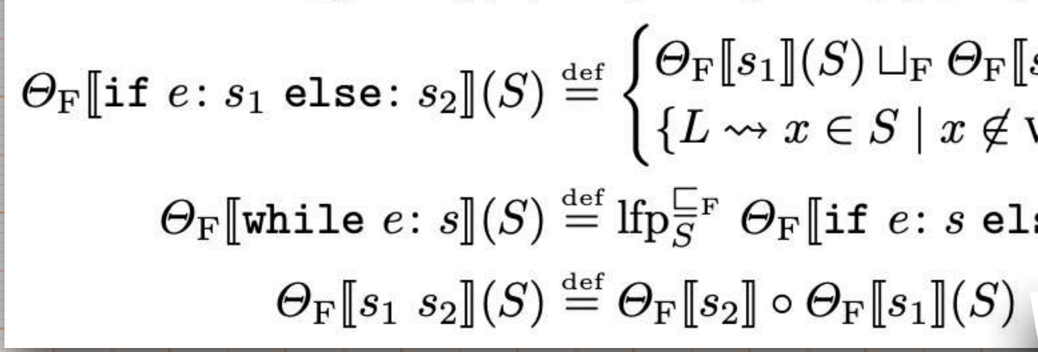
$[\![P]\!]_{\mathrm{F}}$

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

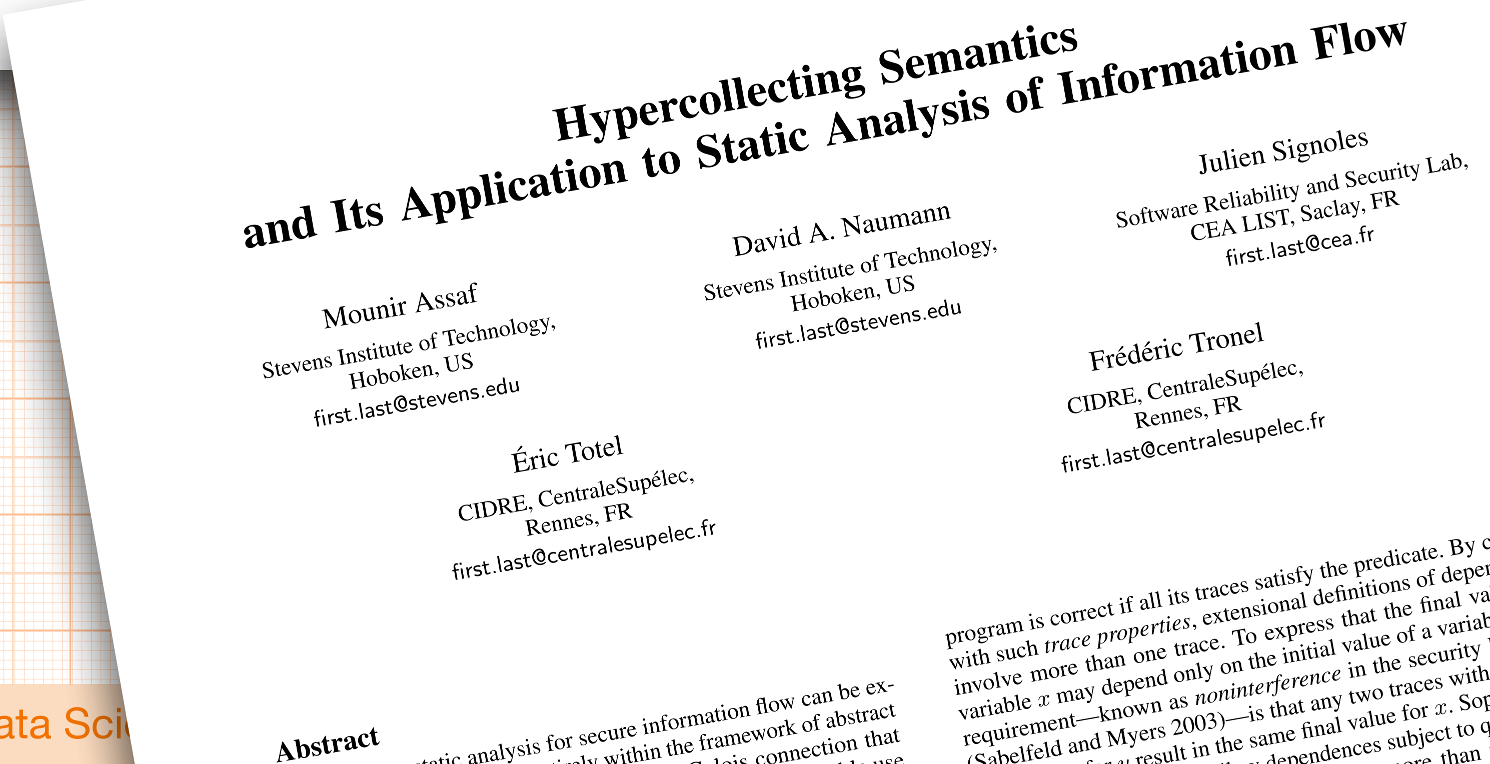
- **input variables** are **high-security variables**
- **output variables** are **low-security variables**



$$e ::= v \mid x \mid \texttt{not}\ e \mid e\ \texttt{and}\ e \mid e\ \texttt{or}\ e \qquad \text{(expressions)}$$
$$s ::= \texttt{skip} \mid x = e \mid \texttt{if}\ e\texttt{:}\ s\ \texttt{else:}\ s \mid \texttt{while}\ e\texttt{:}\ s \mid s\ s \qquad \text{(statements)}$$



$$\Theta_{\mathrm{F}}[\![\texttt{skip}]\!](S) \stackrel{\text{def}}{=} S$$
$$\Theta_{\mathrm{F}}[\![x = e]\!](S) \stackrel{\text{def}}{=} \{L \rightsquigarrow y \in S \mid y \neq x\} \cup \{L \rightsquigarrow x \mid \mathcal{V}_{\mathrm{F}}[\![e]\!]S\}$$
$$\Theta_{\mathrm{F}}[\![\texttt{if}\ e\texttt{:}\ s_1\ \texttt{else:}\ s_2]\!](S) \stackrel{\text{def}}{=} \begin{cases} \Theta_{\mathrm{F}}[\![s_1]\!](S) \sqcup_{\mathrm{F}} \Theta_{\mathrm{F}}[\![s_2]\!](S) & \text{if } \mathcal{V}_{\mathrm{F}}[\![e]\!]S \\ \{L \rightsquigarrow x \in S \mid x \notin \mathrm{W}(s_1) \cup \mathrm{W}(s_2)\} & \text{otherwise} \end{cases}$$
$$\Theta_{\mathrm{F}}[\![\texttt{while}\ e\texttt{:}\ s]\!](S) \stackrel{\text{def}}{=} \mathrm{lfp}_{S}^{\sqsubseteq_{\mathrm{F}}}\ \Theta_{\mathrm{F}}[\![\texttt{if}\ e\texttt{:}\ s\ \texttt{else:}\ \texttt{skip}]\!]$$
$$\Theta_{\mathrm{F}}[\![s_1\ s_2]\!](S) \stackrel{\text{def}}{=} \Theta_{\mathrm{F}}[\![s_2]\!] \circ \Theta_{\mathrm{F}}[\![s_1]\!](S)$$



Hypercollecting Semantics
and Its Application to Static Analysis of Information Flow

Mounir Assaf
Stevens Institute of Technology,
Hoboken, US
first.last@stevens.edu

David A. Naumann
Stevens Institute of Technology,
Hoboken, US
first.last@stevens.edu

Julien Signoles
Software Reliability and Security Lab,
CEA LIST, Saclay, FR
first.last@cea.fr

Éric Totel
CIDRE, CentraleSupélec,
Rennes, FR
first.last@centralesupelec.fr

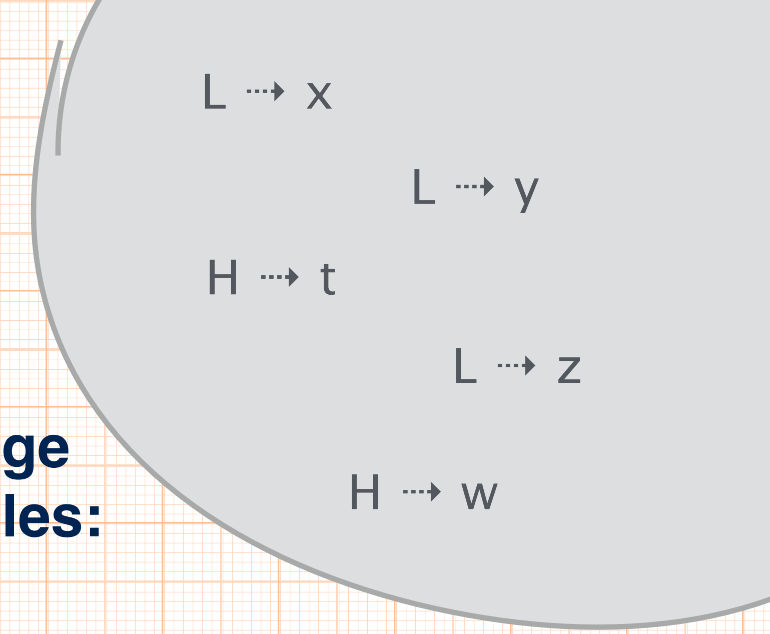
Frédéric Tronel
CIDRE, CentraleSupélec,
Rennes, FR
first.last@centralesupelec.fr

Abstract

tatic analysis for secure information flow can be ex- tirely within the framework of abstract Galois connection that

program is correct if all its traces satisfy the predicate. By c with such *trace properties*, extensional definitions of depe involve more than one trace. To express that the final va variable x may depend only on the initial value of a variab requirement—known as *noninterference* in the security (Sabelfeld and Myers 2003)—is that any two traces with for x result in the same final value for x. Sop dependences subject to q more than

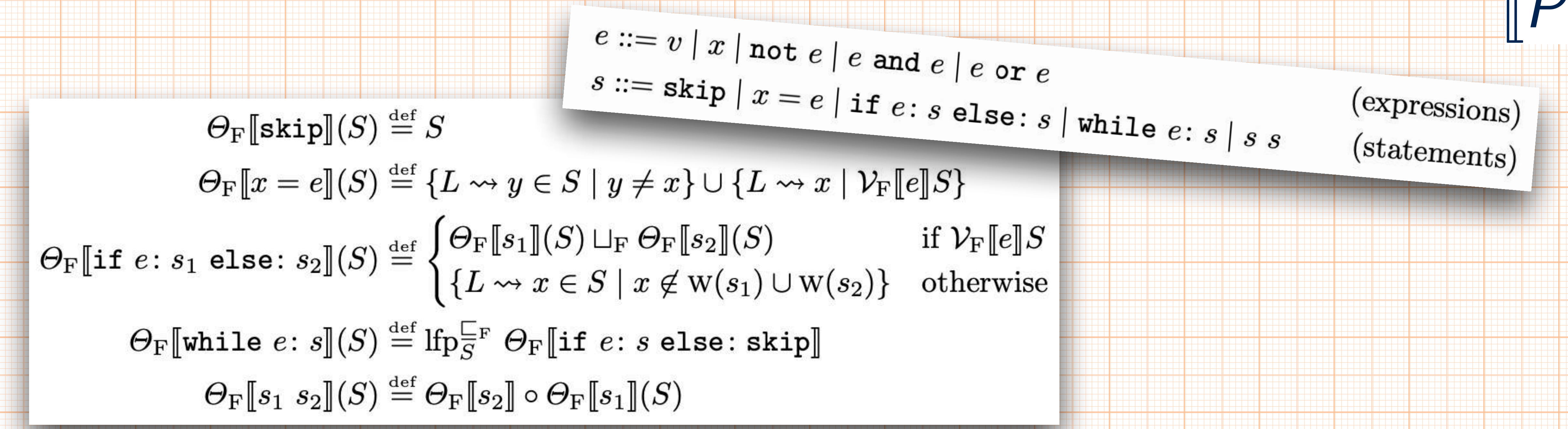
Secure Information Flow



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**

$\llbracket P \rrbracket_{\mathrm{F}}$



$$e ::= v \mid x \mid \texttt{not}\ e \mid e\ \texttt{and}\ e \mid e\ \texttt{or}\ e \qquad \text{(expressions)}$$

$$s ::= \texttt{skip} \mid x = e \mid \texttt{if}\ e\texttt{:}\ s\ \texttt{else:}\ s \mid \texttt{while}\ e\texttt{:}\ s \mid s\ s \qquad \text{(statements)}$$

$$\Theta_{\mathrm{F}}\llbracket \texttt{skip} \rrbracket(S) \overset{\text{def}}{=} S$$

$$\Theta_{\mathrm{F}}\llbracket x = e \rrbracket(S) \overset{\text{def}}{=} \{L \rightsquigarrow y \in S \mid y \neq x\} \cup \{L \rightsquigarrow x \mid \mathcal{V}_{\mathrm{F}}\llbracket e \rrbracket S\}$$

$$\Theta_{\mathrm{F}}\llbracket \texttt{if}\ e\texttt{:}\ s_1\ \texttt{else:}\ s_2 \rrbracket(S) \overset{\text{def}}{=} \begin{cases} \Theta_{\mathrm{F}}\llbracket s_1 \rrbracket(S) \sqcup_{\mathrm{F}} \Theta_{\mathrm{F}}\llbracket s_2 \rrbracket(S) & \text{if } \mathcal{V}_{\mathrm{F}}\llbracket e \rrbracket S \\ \{L \rightsquigarrow x \in S \mid x \notin \mathrm{w}(s_1) \cup \mathrm{w}(s_2)\} & \text{otherwise} \end{cases}$$

$$\Theta_{\mathrm{F}}\llbracket \texttt{while}\ e\texttt{:}\ s \rrbracket(S) \overset{\text{def}}{=} \mathrm{lfp}_{S}^{\sqsubseteq_{\mathrm{F}}}\ \Theta_{\mathrm{F}}\llbracket \texttt{if}\ e\texttt{:}\ s\ \texttt{else:}\ \texttt{skip} \rrbracket$$

$$\Theta_{\mathrm{F}}\llbracket s_1\ s_2 \rrbracket(S) \overset{\text{def}}{=} \Theta_{\mathrm{F}}\llbracket s_2 \rrbracket \circ \Theta_{\mathrm{F}}\llbracket s_1 \rrbracket(S)$$

```
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

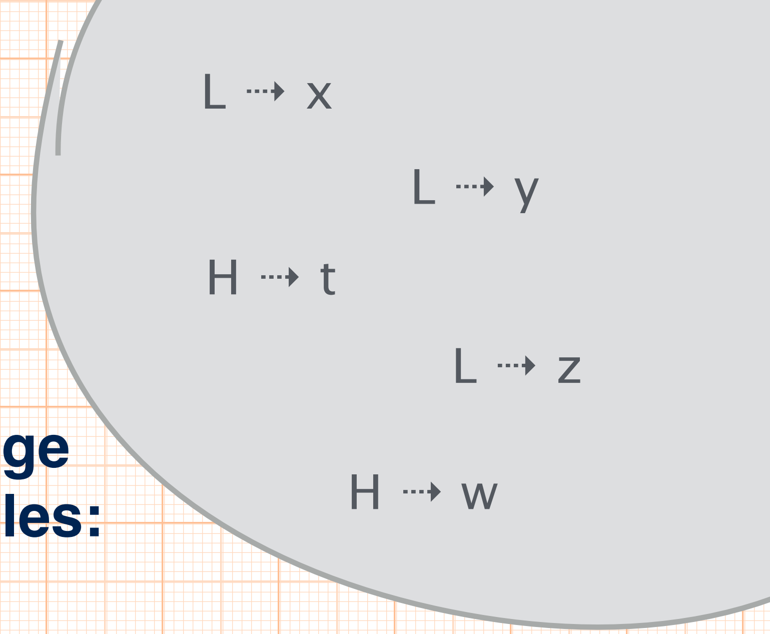
L ⇢ passing, H ⇢ english, math, science, bonus

L ⇢ passing, H ⇢ english, math, science, bonus

H ⇢ english, math, science, bonus, passing

H ⇢ english, math, science, bonus, passing

Secure Information Flow


L ⇢ x

L ⇢ y

H ⇢ t

L ⇢ z

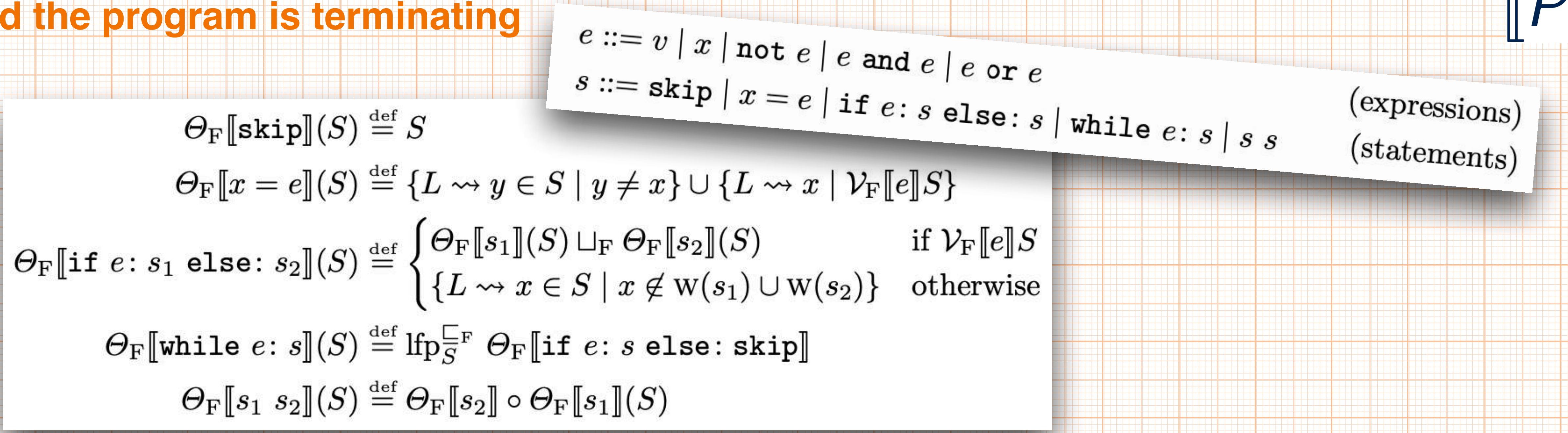
H ⇢ w

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

$[\![P]\!]_{\mathrm{F}}$


$e ::= v \mid x \mid \texttt{not}\ e \mid e\ \texttt{and}\ e \mid e\ \texttt{or}\ e$ (expressions)

$s ::= \texttt{skip} \mid x = e \mid \texttt{if}\ e\texttt{:}\ s\ \texttt{else:}\ s \mid \texttt{while}\ e\texttt{:}\ s \mid s\ s$ (statements)

$$\Theta_{\mathrm{F}}[\![\texttt{skip}]\!](S) \overset{\text{def}}{=} S$$

$$\Theta_{\mathrm{F}}[\![x = e]\!](S) \overset{\text{def}}{=} \{L \rightsquigarrow y \in S \mid y \neq x\} \cup \{L \rightsquigarrow x \mid \mathcal{V}_{\mathrm{F}}[\![e]\!]S\}$$

$$\Theta_{\mathrm{F}}[\![\texttt{if}\ e\texttt{:}\ s_1\ \texttt{else:}\ s_2]\!](S) \overset{\text{def}}{=} \begin{cases} \Theta_{\mathrm{F}}[\![s_1]\!](S) \sqcup_{\mathrm{F}} \Theta_{\mathrm{F}}[\![s_2]\!](S) & \text{if } \mathcal{V}_{\mathrm{F}}[\![e]\!]S \\ \{L \rightsquigarrow x \in S \mid x \notin \mathrm{w}(s_1) \cup \mathrm{w}(s_2)\} & \text{otherwise} \end{cases}$$

$$\Theta_{\mathrm{F}}[\![\texttt{while}\ e\texttt{:}\ s]\!](S) \overset{\text{def}}{=} \mathrm{lfp}_{S}^{\sqsubseteq_{\mathrm{F}}}\ \Theta_{\mathrm{F}}[\![\texttt{if}\ e\texttt{:}\ s\ \texttt{else:}\ \texttt{skip}]\!]$$

$$\Theta_{\mathrm{F}}[\![s_1\ s_2]\!](S) \overset{\text{def}}{=} \Theta_{\mathrm{F}}[\![s_2]\!] \circ \Theta_{\mathrm{F}}[\![s_1]\!](S)$$

```
passing = True
while not english:
    english = False
```

L ⇢ passing, H ⇢ english, math, science, bonus

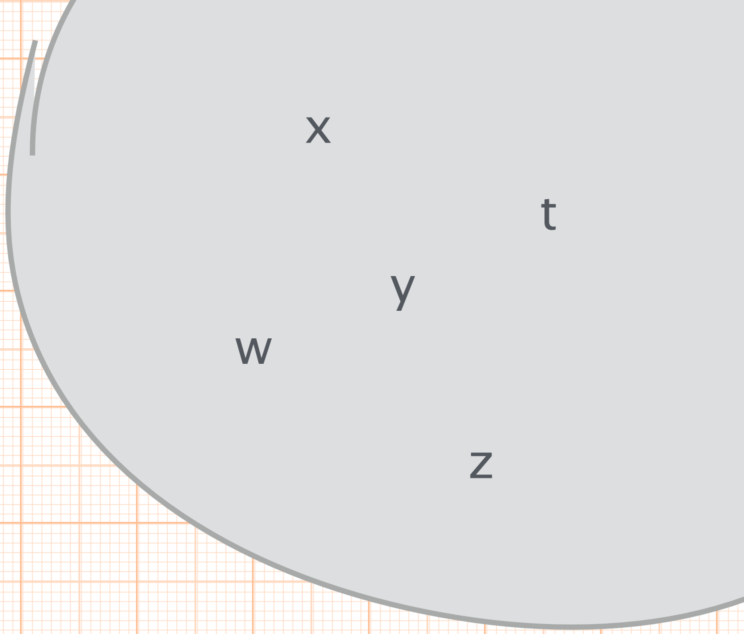
L ⇢ passing, H ⇢ english, math, science, bonus

L ⇢ passing, H ⇢ english, math, science, bonus

Theorem

$$P \models \mathcal{N}^{*}_{I_P} \Leftarrow \gamma_{\rightsquigarrow}(\gamma_F([\![P]\!]_F)) \subseteq \mathcal{N}^{*}_{I_P}$$

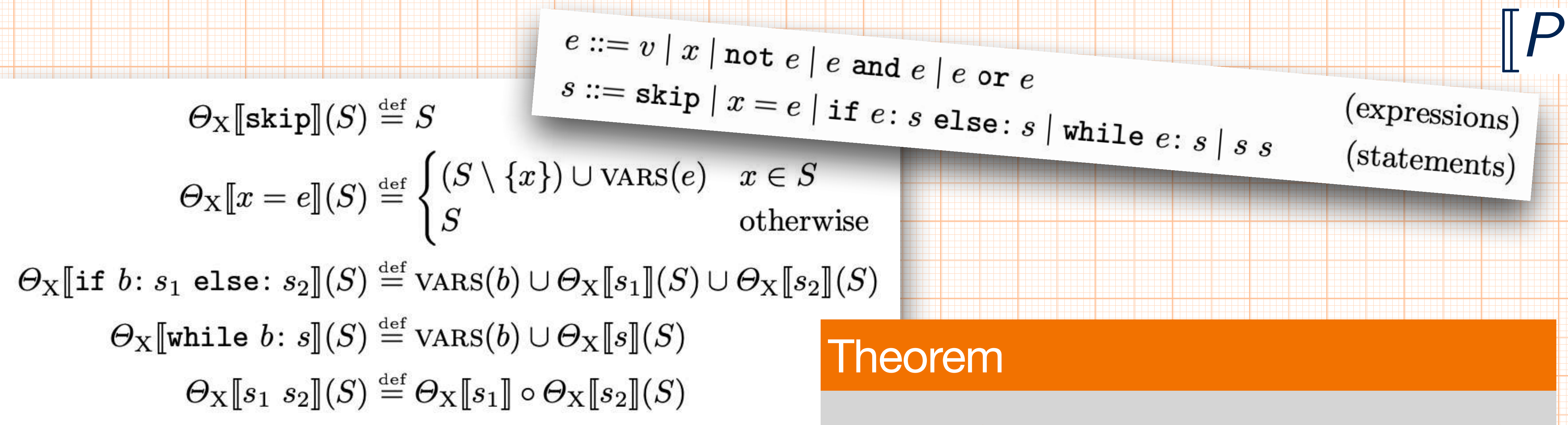
Strong-Liveness



$[\![P]\!]_X$

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**



$e ::= v \mid x \mid \texttt{not}\ e \mid e\ \texttt{and}\ e \mid e\ \texttt{or}\ e$ (expressions)

$s ::= \texttt{skip} \mid x = e \mid \texttt{if}\ e\texttt{:}\ s\ \texttt{else:}\ s \mid \texttt{while}\ e\texttt{:}\ s \mid s\ s$ (statements)

$$\Theta_X[\![\texttt{skip}]\!](S) \overset{\text{def}}{=} S$$

$$\Theta_X[\![x = e]\!](S) \overset{\text{def}}{=} \begin{cases} (S \setminus \{x\}) \cup \textsc{vars}(e) & x \in S \\ S & \text{otherwise} \end{cases}$$

$$\Theta_X[\![\texttt{if}\ b\texttt{:}\ s_1\ \texttt{else:}\ s_2]\!](S) \overset{\text{def}}{=} \textsc{vars}(b) \cup \Theta_X[\![s_1]\!](S) \cup \Theta_X[\![s_2]\!](S)$$

$$\Theta_X[\![\texttt{while}\ b\texttt{:}\ s]\!](S) \overset{\text{def}}{=} \textsc{vars}(b) \cup \Theta_X[\![s]\!](S)$$

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

Theorem

$$P \models \mathcal{N}_J \Leftarrow \gamma_{\rightsquigarrow}(\gamma_X([\![P]\!]_X)) \subseteq \mathcal{N}_J$$

```
passing = True                    ◀·····················  { bonus, math, english }
if not english:                   ◀·····················  { bonus, math, english }
    english = False               ◀·····················  { bonus, math }
if not math:                      ◀·····················  { bonus, math }
    passing = False or bonus      ◀·····················  { bonus, math }
if not math:                      ◀·····················  { bonus, math }
    passing = False or bonus      ◀·····················  { bonus }
                                  ◀·····················  { passing }
```

Syntactic (Non-)Usage

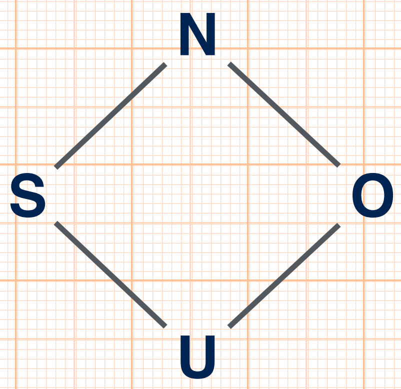
x ⇢ U

y ⇢ S | y ⇢ U

t ⇢ N

z ⇢ N

w ⇢ O | w ⇢ U



- **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

$[\![P]\!]_{\mathrm{U}}$

$$\Theta_{\mathrm{Q}}[\![\texttt{skip}]\!](q) \stackrel{\text{def}}{=} q$$

$$\Theta_{\mathrm{Q}}[\![x = e]\!](q) \stackrel{\text{def}}{=} \mathrm{ASSIGN}[\![x = e]\!](q)$$

$$\begin{aligned}\Theta_{\mathrm{Q}}[\![\texttt{if } b\texttt{: } s_1 \texttt{ else: } s_2]\!](q) \stackrel{\text{def}}{=} & \ \mathrm{POP} \circ \mathrm{FILTER}[\![b]\!] \circ \Theta_{\mathrm{Q}}[\![s_1]\!] \circ \mathrm{PUSH}(q) \\ & \sqcup_{\mathrm{Q}} \mathrm{POP} \circ \mathrm{FILTER}[\![b]\!] \circ \Theta_{\mathrm{Q}}[\![s_2]\!] \circ \mathrm{PUSH}(q)\end{aligned}$$

$$\Theta_{\mathrm{Q}}[\![\texttt{while } b\texttt{: } s]\!](q) \stackrel{\text{def}}{=} \mathrm{lfp}_t^{\sqsubseteq_{\mathrm{Q}}} \ \Theta_{\mathrm{Q}}[\![\texttt{if } b\texttt{: } s \texttt{ else: skip}]\!]$$

$$\Theta_{\mathrm{Q}}[\![s_1 \ s_2]\!](q) \stackrel{\text{def}}{=} \Theta_{\mathrm{Q}}[\![s_1]\!] \circ \Theta_{\mathrm{Q}}[\![s_2]\!](q)$$

```
passing = True

if not english:

    english = False
    ●.......................... math, bonus, passing ⇢ S | math, bonus, passing ⇢ U
●.............................. math, bonus, passing ⇢ U
if not math:
    ●.......................... math ⇢ S, bonus ⇢ U, passing ⇢ O | …
    passing = False or bonus
    ●.......................... math, bonus, passing ⇢ S | math, bonus, passing ⇢ U
●.............................. math, bonus, passing ⇢ U
if not math:
    ●.......................... bonus ⇢ U, passing ⇢ O | passing ⇢ U
    passing = False or bonus
    ●.......................... passing ⇢ S | passing ⇢ U
●.............................. passing ⇢ U
```

Syntactic (Non-)Usage

x ⇢ U

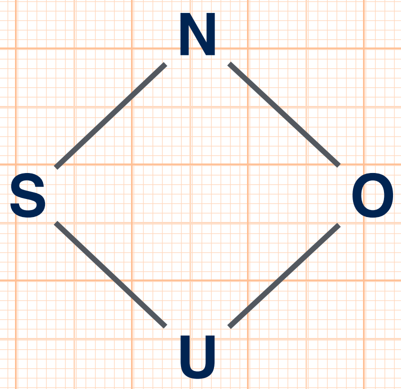
y ⇢ S | y ⇢ U

t ⇢ N

z ⇢ N

w ⇢ O | w ⇢ U

$[\![P]\!]_{\mathrm{U}}$



- **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

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

math, bonus ⇢ U, passing ⇢ O

math, bonus, passing ⇢ U

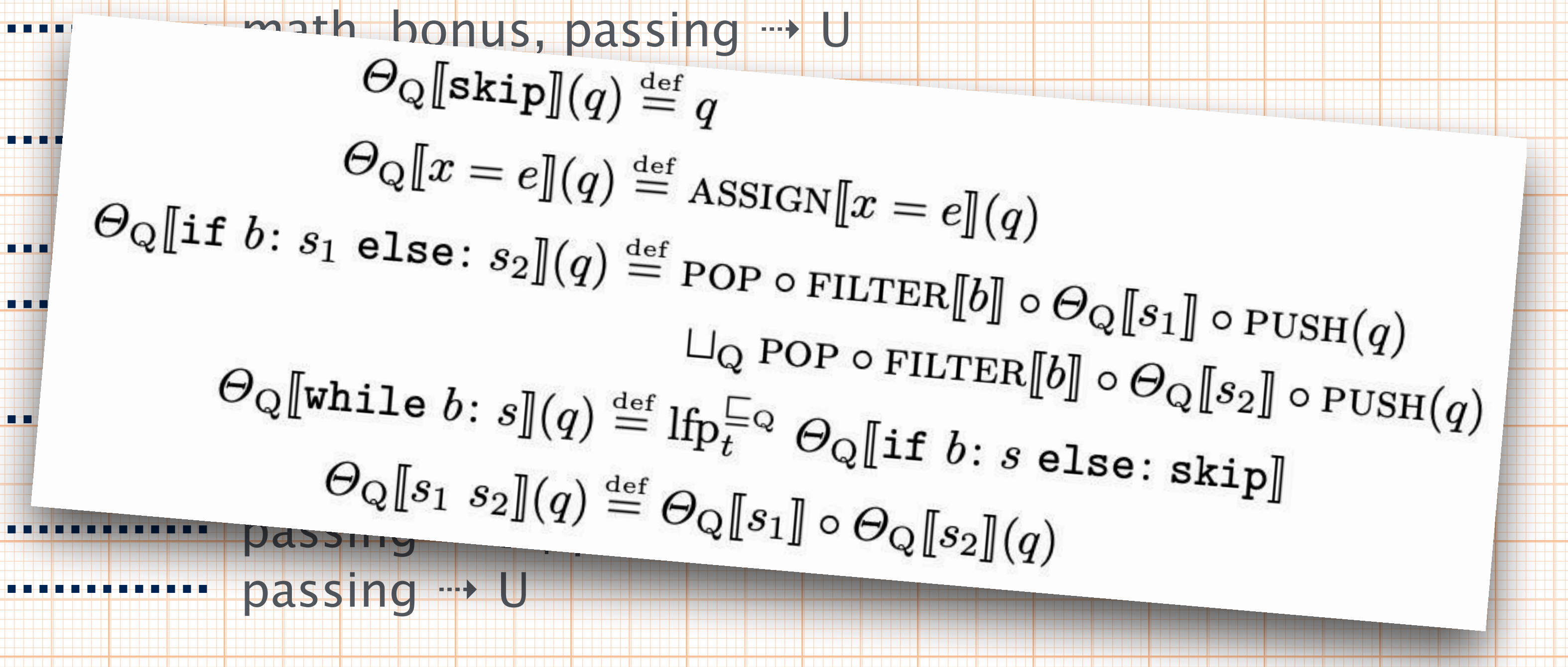
math, bonus, passing ⇢ S | math, bonus, passing ⇢ U

math, bonus, passing ⇢ S | math, bonus, passing ⇢ U

math, bonus, passing ⇢ U

passing

passing ⇢ U



$$\Theta_{\mathrm{Q}}[\![\texttt{skip}]\!](q) \overset{\text{def}}{=} q$$

$$\Theta_{\mathrm{Q}}[\![x = e]\!](q) \overset{\text{def}}{=} \mathrm{ASSIGN}[\![x = e]\!](q)$$

$$\begin{aligned}\Theta_{\mathrm{Q}}[\![\texttt{if } b\texttt{: } s_1 \texttt{ else: } s_2]\!](q) \overset{\text{def}}{=}\ & \mathrm{POP} \circ \mathrm{FILTER}[\![b]\!] \circ \Theta_{\mathrm{Q}}[\![s_1]\!] \circ \mathrm{PUSH}(q) \\ & \sqcup_{\mathrm{Q}} \mathrm{POP} \circ \mathrm{FILTER}[\![b]\!] \circ \Theta_{\mathrm{Q}}[\![s_2]\!] \circ \mathrm{PUSH}(q)\end{aligned}$$

$$\Theta_{\mathrm{Q}}[\![\texttt{while } b\texttt{: } s]\!](q) \overset{\text{def}}{=} \mathrm{lfp}_t^{\sqsubseteq_{\mathrm{Q}}}\ \Theta_{\mathrm{Q}}[\![\texttt{if } b\texttt{: } s \texttt{ else: skip}]\!]$$

$$\Theta_{\mathrm{Q}}[\![s_1\ s_2]\!](q) \overset{\text{def}}{=} \Theta_{\mathrm{Q}}[\![s_1]\!] \circ \Theta_{\mathrm{Q}}[\![s_2]\!](q)$$

Syntactic (Non-)Usage

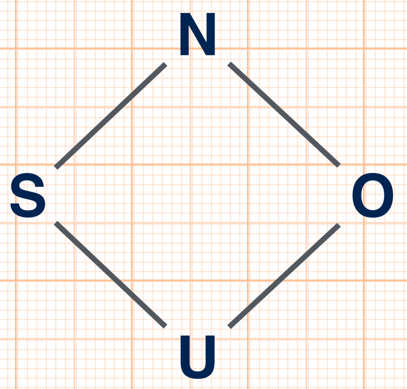
x ⇢ U

y ⇢ S | y ⇢ U

t ⇢ N

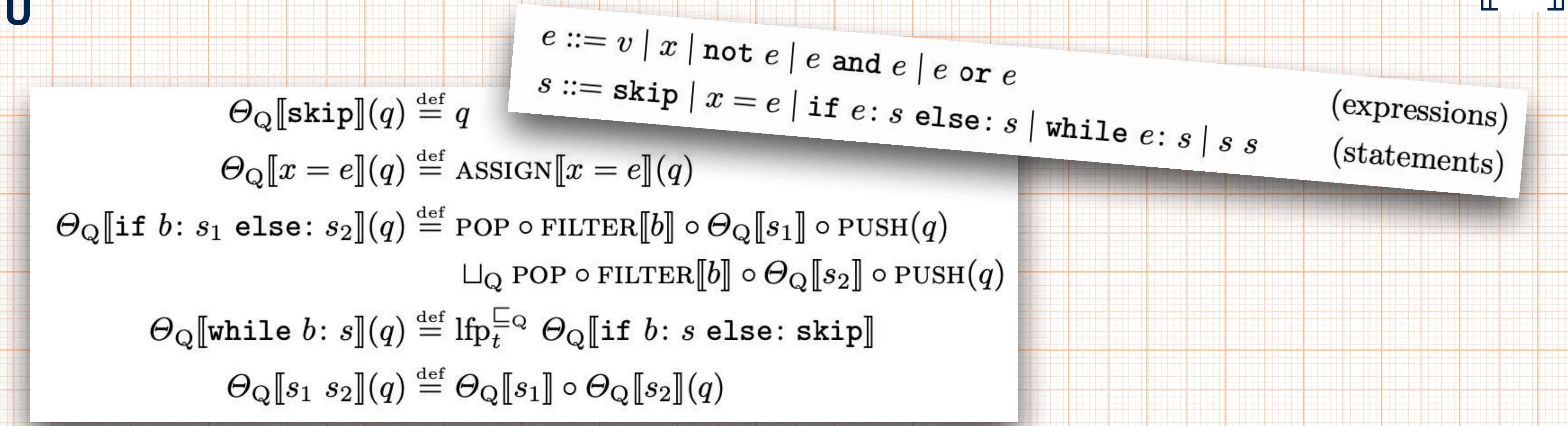
z ⇢ N

w ⇢ O | w ⇢ U



- **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

$[\![P]\!]_{\mathrm{U}}$



$e ::= v \mid x \mid \texttt{not}\ e \mid e\ \texttt{and}\ e \mid e\ \texttt{or}\ e$ (expressions)

$s ::= \texttt{skip} \mid x = e \mid \texttt{if}\ e\texttt{:}\ s\ \texttt{else:}\ s \mid \texttt{while}\ e\texttt{:}\ s \mid s\ s$ (statements)

$$\Theta_{\mathrm{Q}}[\![\texttt{skip}]\!](q) \overset{\text{def}}{=} q$$

$$\Theta_{\mathrm{Q}}[\![x = e]\!](q) \overset{\text{def}}{=} \mathrm{ASSIGN}[\![x = e]\!](q)$$

$$\begin{aligned}\Theta_{\mathrm{Q}}[\![\texttt{if}\ b\texttt{:}\ s_1\ \texttt{else:}\ s_2]\!](q) &\overset{\text{def}}{=} \mathrm{POP} \circ \mathrm{FILTER}[\![b]\!] \circ \Theta_{\mathrm{Q}}[\![s_1]\!] \circ \mathrm{PUSH}(q)\\ &\quad \sqcup_{\mathrm{Q}} \mathrm{POP} \circ \mathrm{FILTER}[\![b]\!] \circ \Theta_{\mathrm{Q}}[\![s_2]\!] \circ \mathrm{PUSH}(q)\end{aligned}$$

$$\Theta_{\mathrm{Q}}[\![\texttt{while}\ b\texttt{:}\ s]\!](q) \overset{\text{def}}{=} \mathrm{lfp}_t^{\sqsubseteq_{\mathrm{Q}}}\ \Theta_{\mathrm{Q}}[\![\texttt{if}\ b\texttt{:}\ s\ \texttt{else:}\ \texttt{skip}]\!]$$

$$\Theta_{\mathrm{Q}}[\![s_1\ s_2]\!](q) \overset{\text{def}}{=} \Theta_{\mathrm{Q}}[\![s_1]\!] \circ \Theta_{\mathrm{Q}}[\![s_2]\!](q)$$

```
passing = True
while not english:
    english = False
```

◀┈┈┈┈┈ passing ⇢ O

◀┈┈┈┈┈ passing ⇢ U

◀┈┈┈┈┈ passing ⇢ U

Theorem

$$P \models \mathcal{N}_J^* \Leftarrow \gamma_{\rightsquigarrow}(\gamma_Q([\![P]\!]_Q)) \subseteq \mathcal{N}_J^*$$

Piecewise Syntactic (Non-)Usage

{0} N {i} U {i+1} O {len}

{0} N {len}

{0} S {len} | {0} U {len}

$e ::= v \mid x \mid \texttt{not}\ e \mid e\ \texttt{and}\ e \mid e\ \texttt{or}\ e \mid a[e] \mid \texttt{len}(a) \mid e \oplus e \mid e \bowtie e$ (expressions)

$s ::= \texttt{skip} \mid x = e \mid \texttt{if}\ e\texttt{:}\ s\ \texttt{else:}\ s \mid \texttt{while}\ e\texttt{:}\ s \mid s\ s \mid a[e] = e$ (statements)

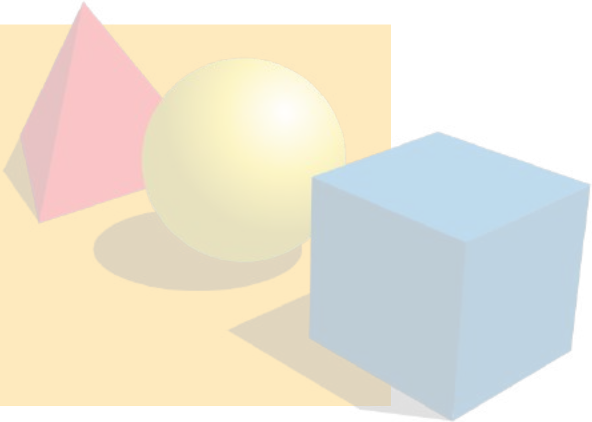
$[\![P]\!]_S$

```
grades = list(map(int, input().split()))

count = 0
●·························· grades ⇢ {0} N {1}? U {2}? U {len(grades)}?
i = 1 ◀····················· ERROR: 1 SHOULD BE 0
●·························· grades ⇢ {0} N {i}? U {i+1}? U {len(grades)}?
while i < len(grades):
    ●······················ grades ⇢ {0} N {i}? U {i+1}? S {i+2}? S {len(grades)}? | …
    if grades[i] < 4:
        count = count + 1
        ●·················· grades ⇢ {0} N {i+1}? S {i+2}? S {len(grades)}? | … | …
    ●······················ grades ⇢ {0} N {i+1}? S {i+2}? S {len(grades)}? | …
    i = i + 1
    ●······················ grades ⇢ {0} N {i}? S {i+1}? S {len(grades)}? | …
●·························· grades ⇢ {0} N {len(grades)}?
if 2 * count < len(grades):
    passing = True
else:
    passing = False
●·························· grades ⇢ { 0 } N { len(grades) }?
print(passing)
```

Unused Data Analysis

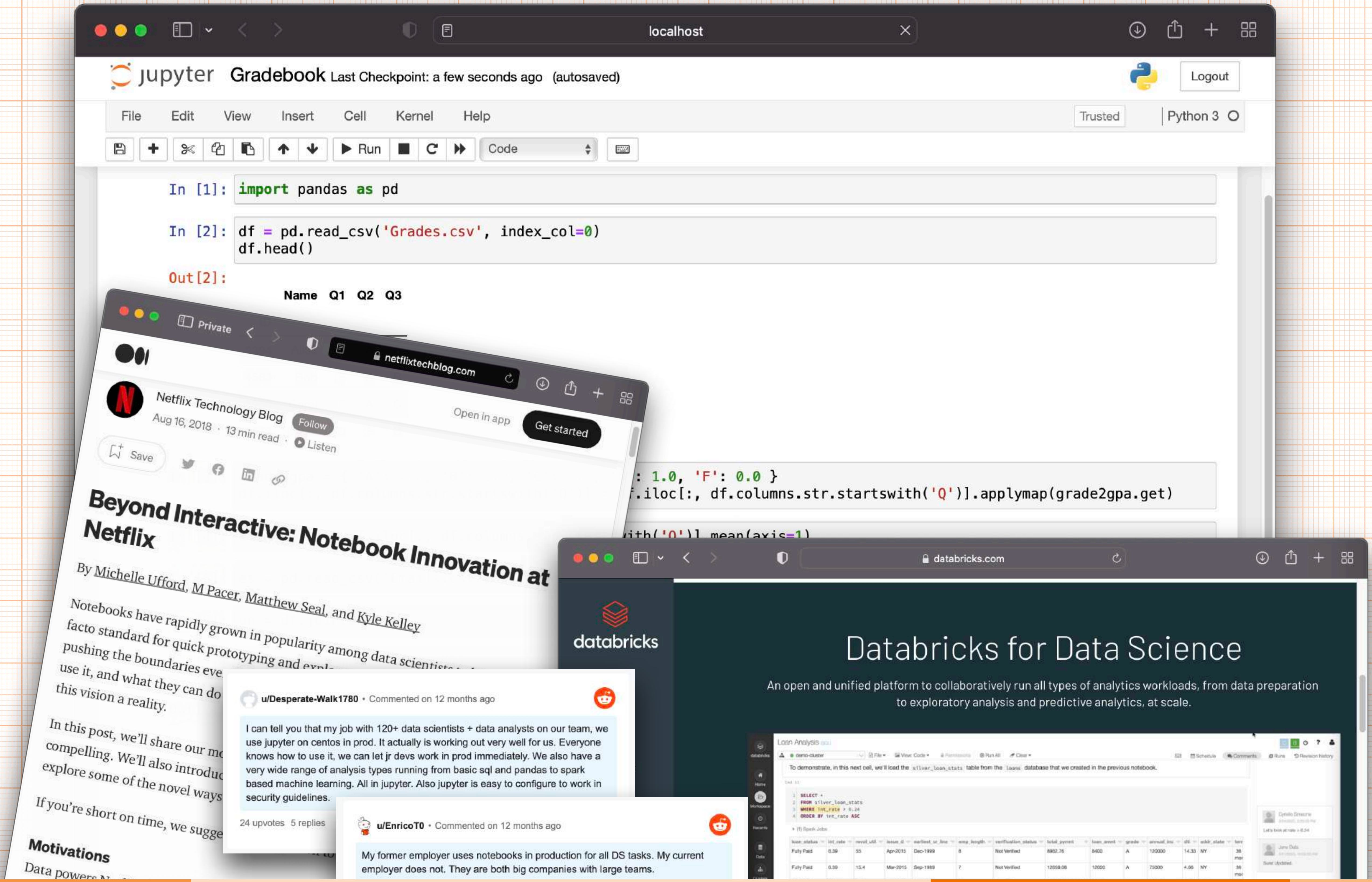
practical tools
targeting specific programs

algorithmic approaches
to decide program properties



mathematical models
of the program behavior

Jupyter Notebooks



```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('Grades.csv', index_col=0)
        df.head()

Out[2]:
        Name  Q1  Q2  Q3
```

```
: 1.0, 'F': 0.0 }
.iloc[:, df.columns.str.startswith('Q')].applymap(grade2gpa.get)
ith('Q')].mean(axis=1)
```

Beyond Interactive: Notebook Innovation at Netflix

By Michelle Ufford, M Pacer, Matthew Seal, and Kyle Kelley

Notebooks have rapidly grown in popularity among data scientists … facto standard for quick prototyping and expl… pushing the boundaries eve… use it, and what they can do… this vision a reality.

In this post, we'll share our m… compelling. We'll also introduc… explore some of the novel ways…

If you're short on time, we sugge…

Motivations

Data powers N…

u/Desperate-Walk1780 · Commented on 12 months ago

I can tell you that my job with 120+ data scientists + data analysts on our team, we use jupyter on centos in prod. It actually is working out very well for us. Everyone knows how to use it, we can let jr devs work in prod immediately. We also have a very wide range of analysis types running from basic sql and pandas to spark based machine learning. All in jupyter. Also jupyter is easy to configure to work in security guidelines.

24 upvotes 5 replies

u/EnricoT0 · Commented on 12 months ago

My former employer uses notebooks in production for all DS tasks. My current employer does not. They are both big companies with large teams.

Databricks for Data Science

An open and unified platform to collaboratively run all types of analytics workloads, from data preparation to exploratory analysis and predictive analytics, at scale.

Jupyter Notebooks

```
[1] d = genfromtxt('data.csv')

[2] selector = SelectKBest(k=25)
    x = selector.fit_transform(d)

[3] x = genfromtxt('data2.csv')

[4] x_train, x_test, y_train, y_test =
      train_test_split(x, ...)

[5] lr = LogisticRegression()
    lr.fit(x_train, y_train)
    y_pred = lr.predict(x_test)
```

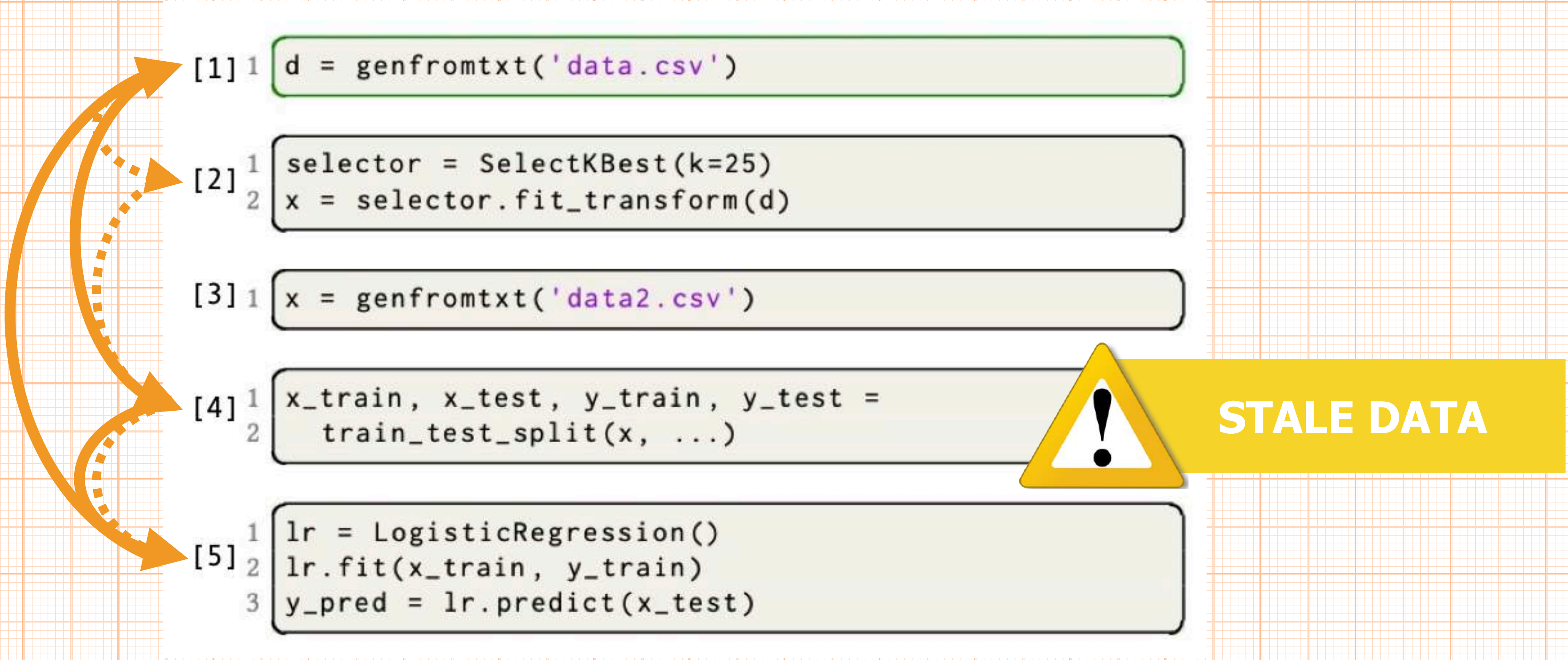
P. Subotić et al. - A Static Analysis Framework for Data Science Notebooks (ICSE 2022)

Jupyter Notebooks

```
[1] d = genfromtxt('data.csv')

[2] selector = SelectKBest(k=25)
    x = selector.fit_transform(d)

[3] x = genfromtxt('data2.csv')

[4] x_train, x_test, y_train, y_test =
      train_test_split(x, ...)

[5] lr = LogisticRegression()
    lr.fit(x_train, y_train)
    y_pred = lr.predict(x_test)
```

P. Subotić et al. - A Static Analysis Framework for Data Science Notebooks (ICSE 2022)

Jupyter Notebooks



```
[1] d = genfromtxt('data.csv')

[2] selector = SelectKBest(k=25)
    x = selector.fit_transform(d)

[3] x = genfromtxt('data2.csv')

[4] x_train, x_test, y_train, y_test =
      train_test_split(x, ...)

[5] lr = LogisticRegression()
    lr.fit(x_train, y_train)
    y_pred = lr.predict(x_test)
```

STALE DATA

P. Subotić et al. - A Static Analysis Framework for Data Science Notebooks (ICSE 2022)

Unexpected Data

Unexpected Data

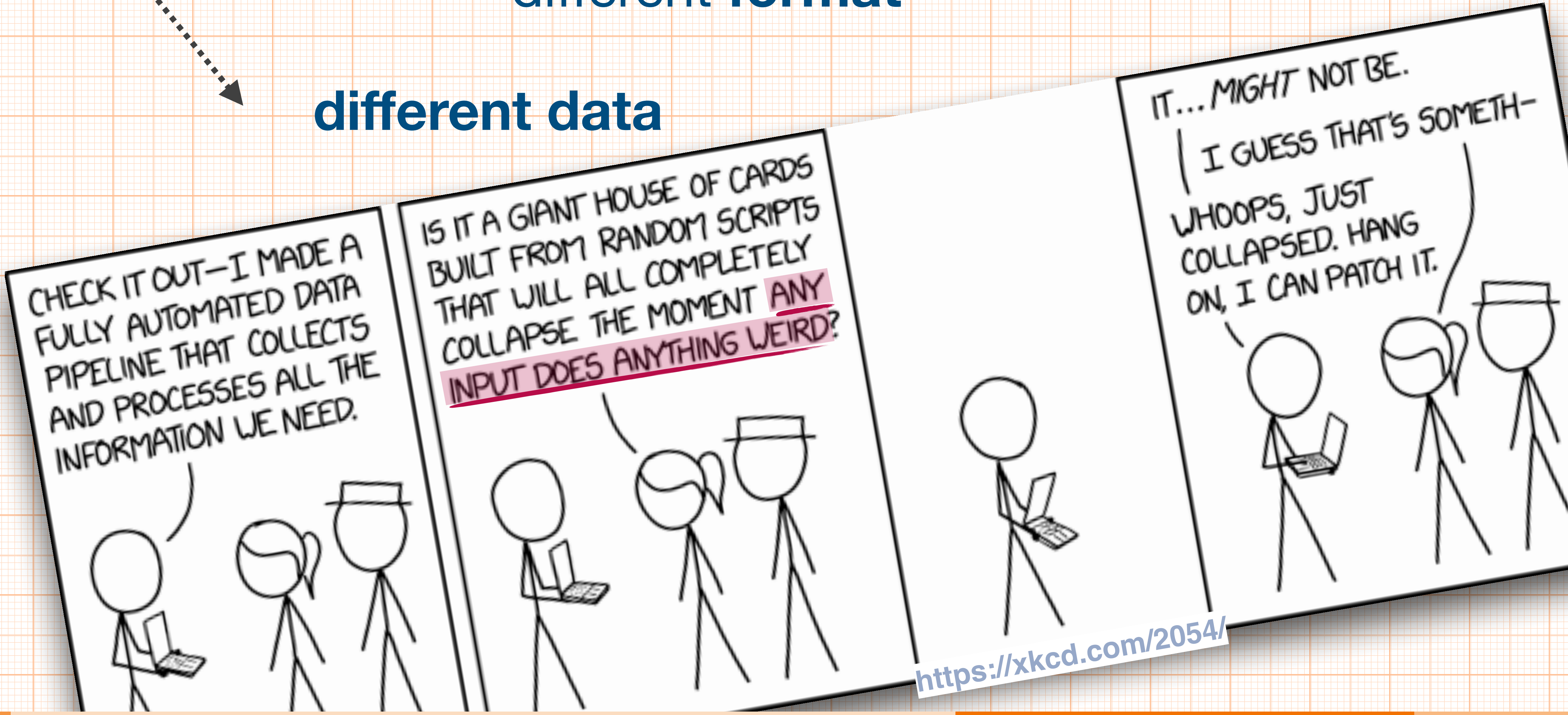
missing values

extra values

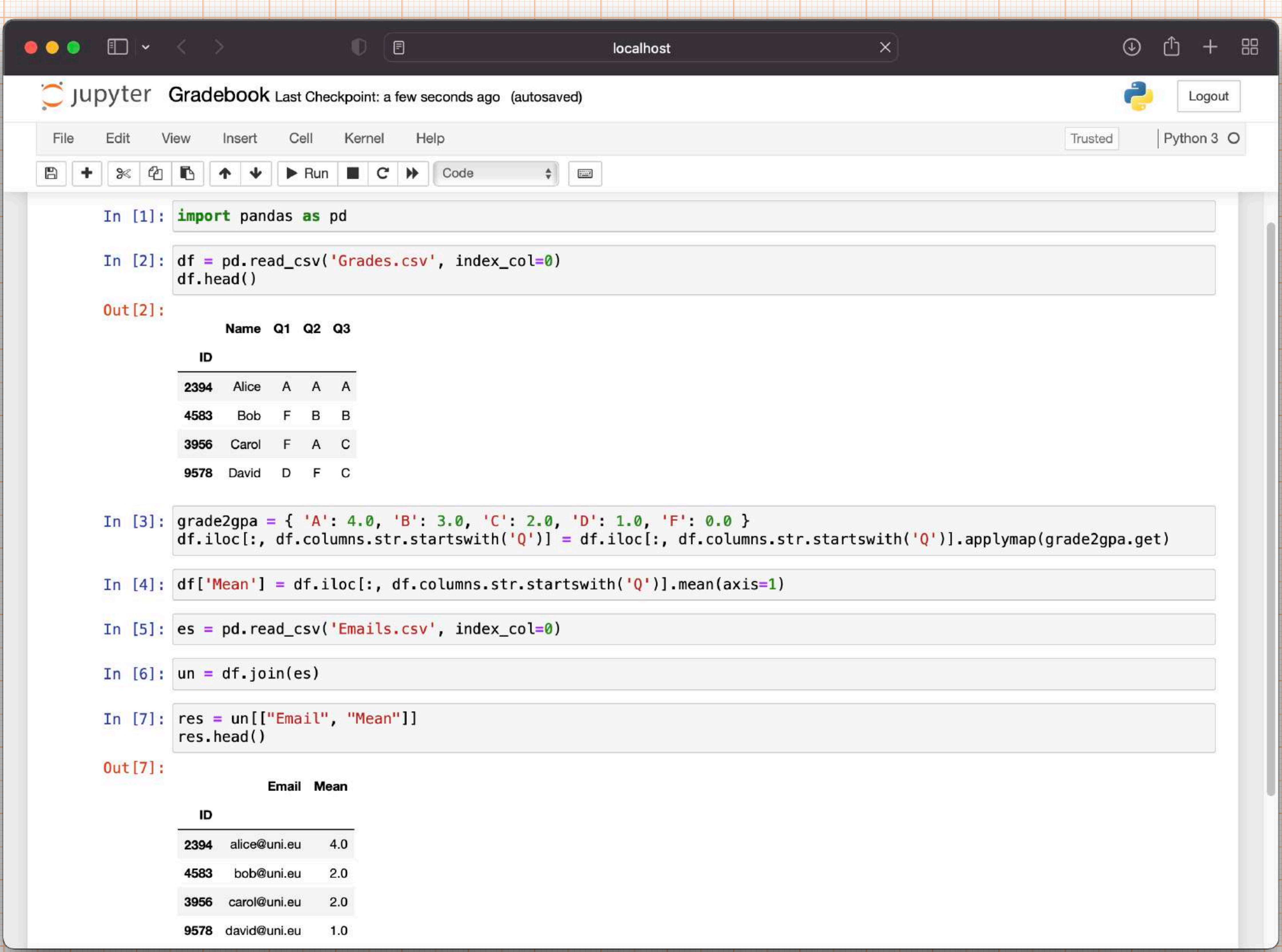
duplicate values

different **format**

different data



https://xkcd.com/2054/



In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('Grades.csv', index_col=0)
df.head()
```

Out[2]:

	Name	Q1	Q2	Q3
ID				
2394	Alice	A	A	A
4583	Bob	F	B	B
3956	Carol	F	A	C
9578	David	D	F	C

In [3]:
```
grade2gpa = { 'A': 4.0, 'B': 3.0, 'C': 2.0, 'D': 1.0, 'F': 0.0 }
df.iloc[:, df.columns.str.startswith('Q')] = df.iloc[:, df.columns.str.startswith('Q')].applymap(grade2gpa.get)
```

In [4]:
```
df['Mean'] = df.iloc[:, df.columns.str.startswith('Q')].mean(axis=1)
```

In [5]:
```
es = pd.read_csv('Emails.csv', index_col=0)
```

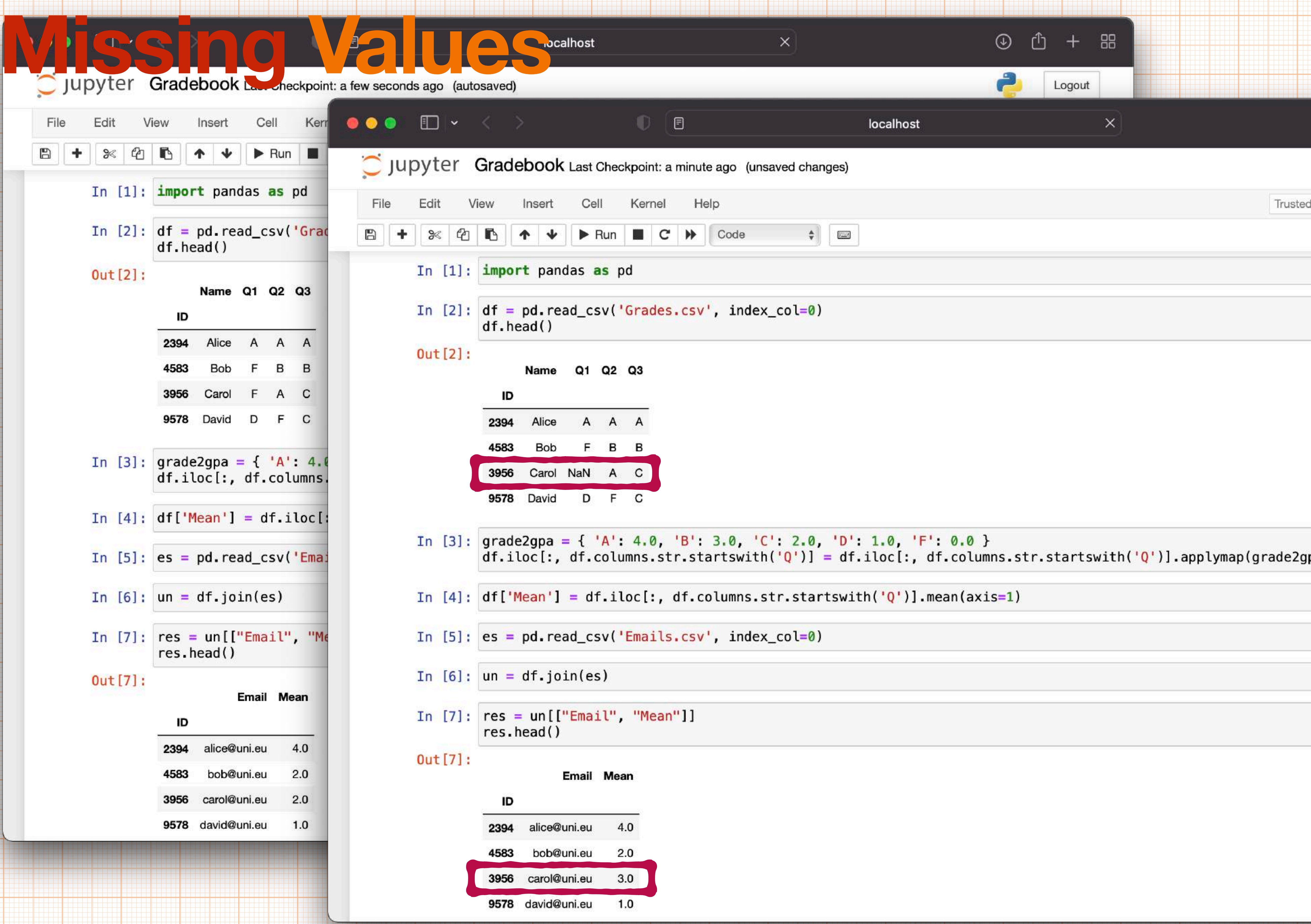
In [6]:
```
un = df.join(es)
```

In [7]:
```
res = un[["Email", "Mean"]]
res.head()
```

Out[7]:

	Email	Mean
ID		
2394	alice@uni.eu	4.0
4583	bob@uni.eu	2.0
3956	carol@uni.eu	2.0
9578	david@uni.eu	1.0

Missing Values



```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('Grac
        df.head()

Out[2]:
```

	Name	Q1	Q2	Q3
ID				
2394	Alice	A	A	A
4583	Bob	F	B	B
3956	Carol	F	A	C
9578	David	D	F	C

```
In [3]: grade2gpa = { 'A': 4.0
        df.iloc[:, df.columns.

In [4]: df['Mean'] = df.iloc[:

In [5]: es = pd.read_csv('Emai

In [6]: un = df.join(es)

In [7]: res = un[["Email", "Me
        res.head()

Out[7]:
```

	Email	Mean
ID		
2394	alice@uni.eu	4.0
4583	bob@uni.eu	2.0
3956	carol@uni.eu	2.0
9578	david@uni.eu	1.0

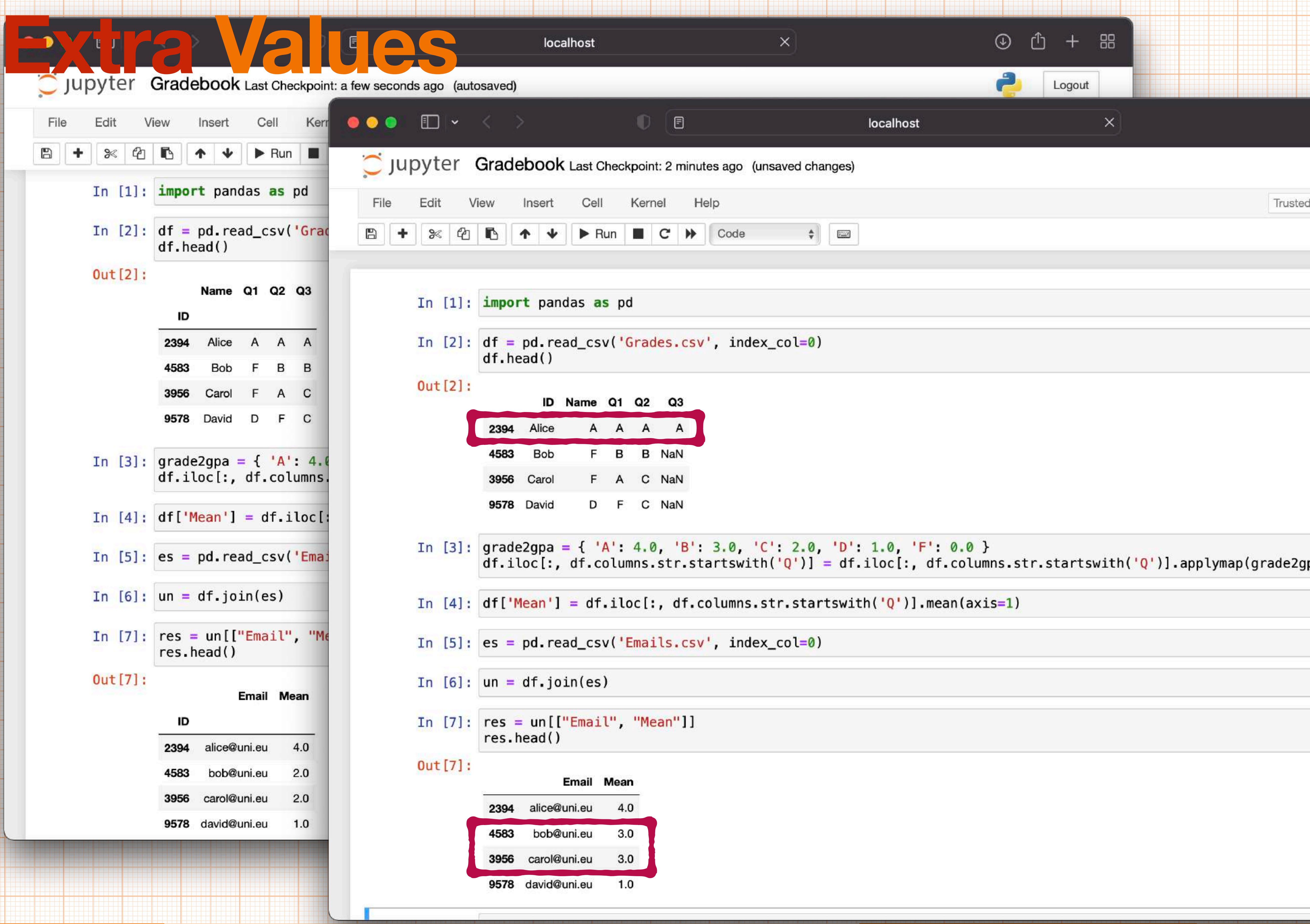
```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('Grades.csv', index_col=0)
        df.head()

Out[2]:
```

	Name	Q1	Q2	Q3
ID				
2394	Alice	A	A	A
4583	Bob	F	B	B
3956	Carol	NaN	A	C
9578	David	D	F	C

```
In [3]: grade2gpa = { 'A': 4.0, 'B': 3.0, 'C': 2.0, 'D': 1.0, 'F': 0.0 }
        df.iloc[:, df.columns.str.startswith('Q')] = df.iloc[:, df.columns.str.startswith('Q')].applymap(grade2gp

In [4]: df['Mean'] = df.iloc[:, df.columns.str.startswith('Q')].mean(axis=1)

In [5]: es = pd.read_csv('Emails.csv', index_col=0)

In [6]: un = df.join(es)

In [7]: res = un[["Email", "Mean"]]
        res.head()

Out[7]:
```

	Email	Mean
ID		
2394	alice@uni.eu	4.0
4583	bob@uni.eu	2.0
3956	carol@uni.eu	3.0
9578	david@uni.eu	1.0

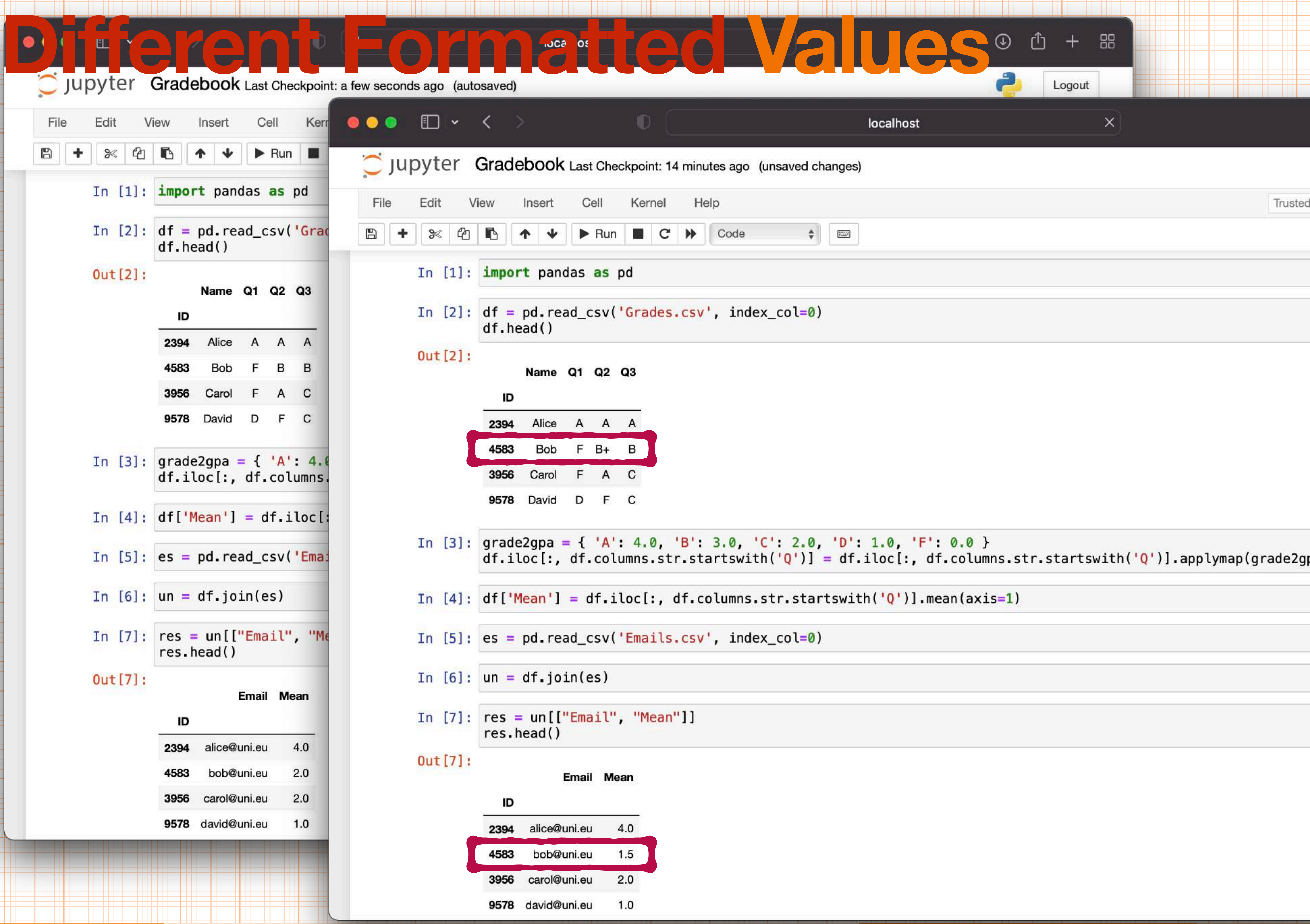
Extra Values



```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('Grac
        df.head()

Out[2]:
              Name  Q1  Q2  Q3
          ID
        2394  Alice  A   A   A
        4583    Bob  F   B   B
        3956  Carol  F   A   C
        9578  David  D   F   C

In [3]: grade2gpa = { 'A': 4.(
        df.iloc[:, df.columns.

In [4]: df['Mean'] = df.iloc[:

In [5]: es = pd.read_csv('Emai

In [6]: un = df.join(es)

In [7]: res = un[["Email", "Me
        res.head()

Out[7]:
                     Email  Mean
          ID
        2394  alice@uni.eu   4.0
        4583    bob@uni.eu   2.0
        3956  carol@uni.eu   2.0
        9578  david@uni.eu   1.0
```

```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('Grades.csv', index_col=0)
        df.head()

Out[2]:
            ID  Name  Q1  Q2   Q3
        2394  Alice    A   A   A    A
        4583    Bob    F   B   B  NaN
        3956  Carol    F   A   C  NaN
        9578  David    D   F   C  NaN

In [3]: grade2gpa = { 'A': 4.0, 'B': 3.0, 'C': 2.0, 'D': 1.0, 'F': 0.0 }
        df.iloc[:, df.columns.str.startswith('Q')] = df.iloc[:, df.columns.str.startswith('Q')].applymap(grade2gp

In [4]: df['Mean'] = df.iloc[:, df.columns.str.startswith('Q')].mean(axis=1)

In [5]: es = pd.read_csv('Emails.csv', index_col=0)

In [6]: un = df.join(es)

In [7]: res = un[["Email", "Mean"]]
        res.head()

Out[7]:
                     Email  Mean
        2394  alice@uni.eu   4.0
        4583    bob@uni.eu   3.0
        3956  carol@uni.eu   3.0
        9578  david@uni.eu   1.0
```

Different Formatted Values



Gradebook Last Checkpoint: a few seconds ago (autosaved)

```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('Grac
        df.head()

Out[2]:
```

	Name	Q1	Q2	Q3
ID				
2394	Alice	A	A	A
4583	Bob	F	B	B
3956	Carol	F	A	C
9578	David	D	F	C

```
In [3]: grade2gpa = { 'A': 4.(
        df.iloc[:, df.columns.

In [4]: df['Mean'] = df.iloc[:

In [5]: es = pd.read_csv('Emai

In [6]: un = df.join(es)

In [7]: res = un[["Email", "Me
        res.head()

Out[7]:
```

	Email	Mean
ID		
2394	alice@uni.eu	4.0
4583	bob@uni.eu	2.0
3956	carol@uni.eu	2.0
9578	david@uni.eu	1.0

Gradebook Last Checkpoint: 14 minutes ago (unsaved changes)

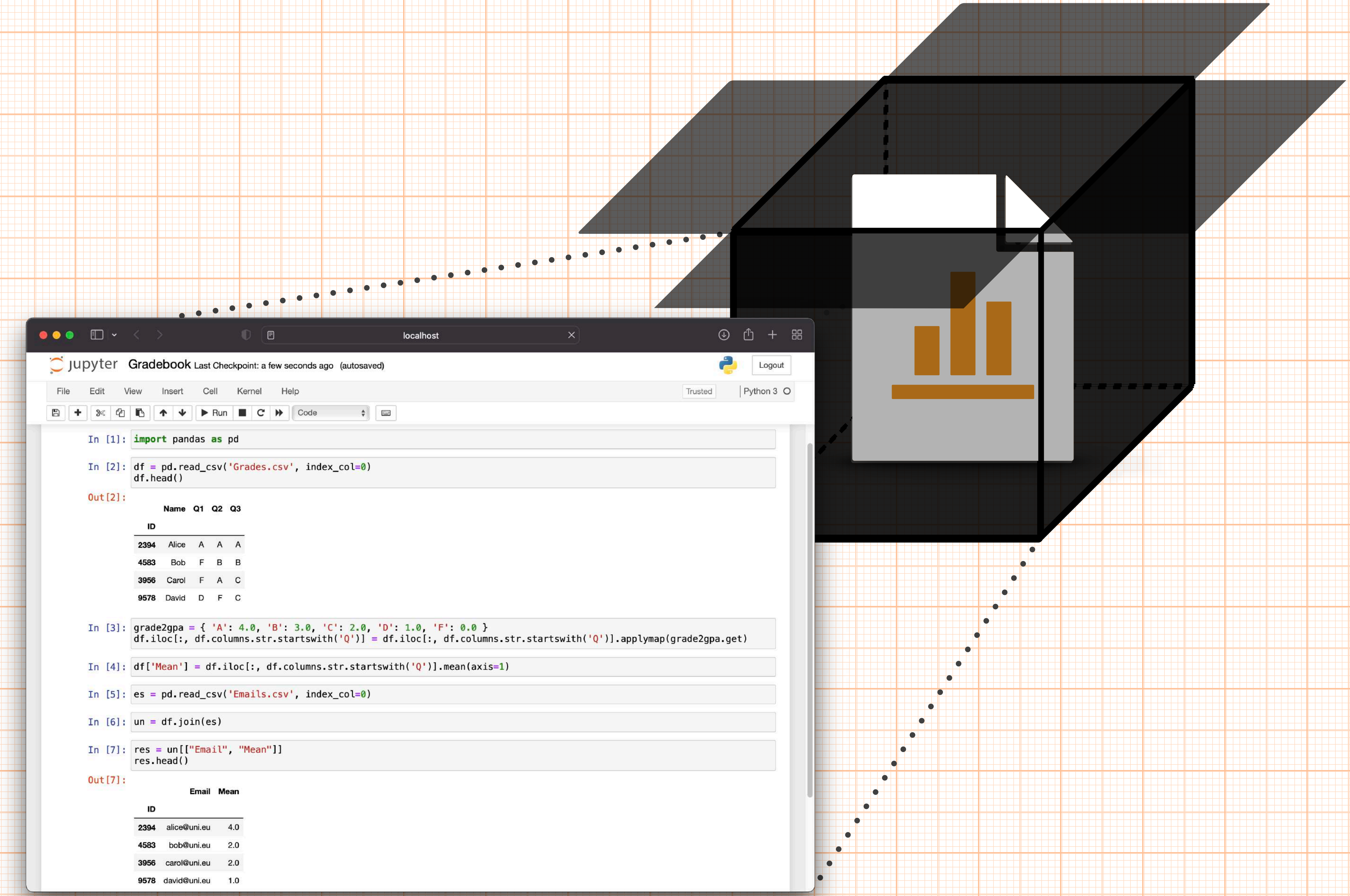
```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('Grades.csv', index_col=0)
        df.head()

Out[2]:
```

	Name	Q1	Q2	Q3
ID				
2394	Alice	A	A	A
4583	Bob	F	B+	B
3956	Carol	F	A	C
9578	David	D	F	C

```
In [3]: grade2gpa = { 'A': 4.0, 'B': 3.0, 'C': 2.0, 'D': 1.0, 'F': 0.0 }
        df.iloc[:, df.columns.str.startswith('Q')] = df.iloc[:, df.columns.str.startswith('Q')].applymap(grade2gp

In [4]: df['Mean'] = df.iloc[:, df.columns.str.startswith('Q')].mean(axis=1)

In [5]: es = pd.read_csv('Emails.csv', index_col=0)

In [6]: un = df.join(es)

In [7]: res = un[["Email", "Mean"]]
        res.head()

Out[7]:
```

	Email	Mean
ID		
2394	alice@uni.eu	4.0
4583	bob@uni.eu	1.5
3956	carol@uni.eu	2.0
9578	david@uni.eu	1.0

Data Expectations Analysis

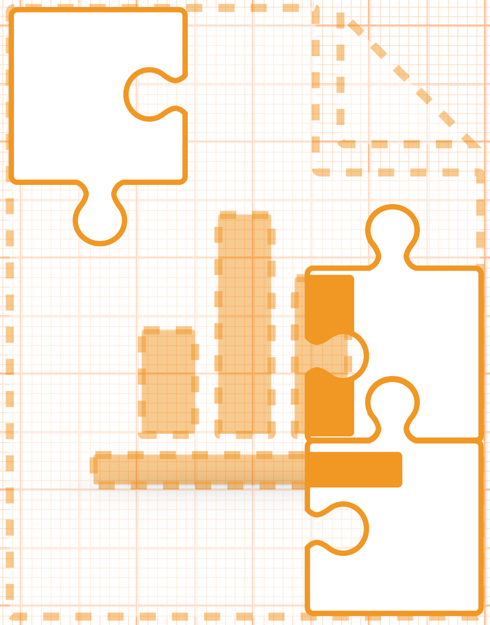


```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('Grades.csv', index_col=0)
        df.head()

Out[2]:
```

ID	Name	Q1	Q2	Q3
2394	Alice	A	A	A
4583	Bob	F	B	B
3956	Carol	F	A	C
9578	David	D	F	C

```
In [3]: grade2gpa = { 'A': 4.0, 'B': 3.0, 'C': 2.0, 'D': 1.0, 'F': 0.0 }
        df.iloc[:, df.columns.str.startswith('Q')] = df.iloc[:, df.columns.str.startswith('Q')].applymap(grade2gpa.get)

In [4]: df['Mean'] = df.iloc[:, df.columns.str.startswith('Q')].mean(axis=1)

In [5]: es = pd.read_csv('Emails.csv', index_col=0)

In [6]: un = df.join(es)

In [7]: res = un[["Email", "Mean"]]
        res.head()

Out[7]:
```

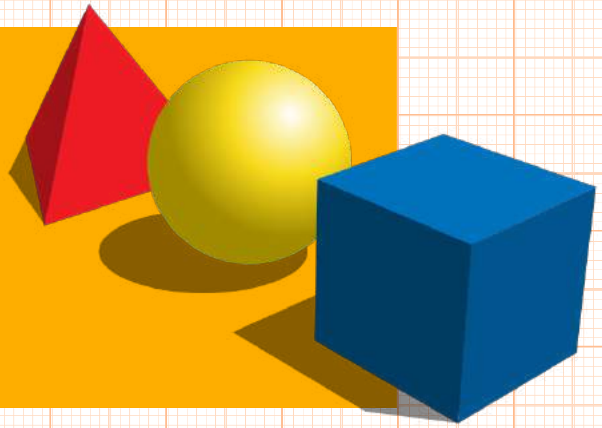
ID	Email	Mean
2394	alice@uni.eu	4.0
4583	bob@uni.eu	2.0
3956	carol@uni.eu	2.0
9578	david@uni.eu	1.0



Data Expectations Analysis

practical tools
targeting specific programs

algorithmic approaches
to decide program properties



mathematical models
of the program behavior

Bibliography

[Urban18] **Caterina Urban and Peter Müller**. An Abstract Interpretation Framework for Input Data Usage. In ESOP, 2018.

[Assaf17] **Mounir Assaf, David A. Naumann, Julien Signoles, Éric Totel, and Frédéric Tronel**. Hypercollecting Semantics and Its Application to Static Analysis of Information Flow. In POPL, 2017.

[Giegerich81] **Robert Giegerich, Ulrich Möncke, and Reinhard Wilhelm**. Invariance of Approximate Semantics with Respect to Program Transformations. 1981.

[Urban19] **Caterina Urban**. Static Analysis of Data Science Software. In SAS, 2019.

[Urban20] **Caterina Urban**. What Programs Want: Automatic Inference of Input Data Specifications. https://arxiv.org/abs/2007.10688, 2020.