A Guided Tour of a Static Analyzer for Data Science Software

Caterina Urban
ANTIQUE Research Team
INRIA & École Normale Supérieure, Paris, France

WIRED

BUSINESS MORE SIGN IN SUBSCRIBE

ERIC NIILER BUSINESS 03.25.2019 07:00 AM

Can AI Be a Fair Judge in Court? Estonia Thinks So

Estonia plans to use an artificial intelligence program to decide some small-claims cases, part of a push to make government services smarter.

The Telegraph

AI used for first time in job interviews in UK to find best applicants

By **Charles Hymas**

27 SEPTEMBER 2019 • 10:00 PM

4 ways to check for skin cancer with your smartphone

Your phone can help you recognize suspicious moles and marks, but you should still see a dermatologist or doctor.

The AI doctor will see you now

Medicine is at the point computer-driven financial trading was in the early 2000s

BROOKE MASTERS

Deep Neural Network Compression for Aircraft Collision Avoidance Systems

Kyle D. Julian[1] and Mykel J. Kochenderfer[2] and Michael P. Owen[3]

***Abstract*—One approach to designing decision making logic for an aircraft collision avoidance system frames the problem as a Markov decision process and optimizes the system using dynamic programming. The resulting collision avoidance strategy can be represented as a numeric table. This methodology has been used in the development of the Airborne Collision Avoidance System X (ACAS X) family of collision avoidance systems for manned and unmanned aircraft, but the high dimensionality of the state space leads to very large tables. To improve storage efficiency, a deep neural network is used to approximate the table. With the use of an asymmetric loss function and a gradient descent algorithm, the parameters for this network can be trained to provide accurate estimates of table values while preserving the relative preferences of the possible advisories for each state. By training multiple networks to represent subtables, the network also decreases the required runtime for computing the collision avoidance advisory. Simulation studies show that the network improves the safety and efficiency of the collision avoidance system. Because only the network parameters need to be stored, the required storage space is reduced by a factor of 1000, enabling the collision avoidance system to operate using current avionics systems.**

I. Introduction

Decades of research have explored a variety of approaches to designing decision making logic for aircraft collision avoidance systems for both manned and unmanned aircraft [1]. Recent work on formulating the problem of collision avoidance as a partially observable Markov decision process (POMDP) has led to the development of the Airborne Collision Avoidance System X (ACAS X) family of collision avoidance systems [2], [3], [4]. The version for manned aircraft, ACAS

floating point storage. A simple technique to reduce the size of the score table is to downsample the table after dynamic programming. To minimize the degradation in decision quality, states are removed in areas where the variation between values in the table are smooth. The downsampling reduces the size of the table by a factor of 180 from that produced by dynamic programming. For the rest of this paper, the downsampled ACAS Xu horizontal table is referred to as the baseline, original table.

Even after downsampling, the current table requires over 2GB of floating point storage, too large for certified avionics systems [6]. Although modern hardware can handle 2GB of storage, the certification process for aircraft computer hardware is expensive and time-consuming, so a solution capable of running on legacy hardware is desired [7]. While there is no formal limit for floating point storage on legacy avionics, a representation occupying less than 120MB would be sufficient.

For an earlier version of ACAS Xa, block compression was introduced to take advantage of the fact that, for many discrete states, the scores for the available actions are identical [8]. One critical contribution of that work was the observation that the table could be stored in IEEE half-precision with no appreciable loss of performance. Block compression was adequate for the ACAS Xa tables that limit advisories to vertical maneuvers, but the ACAS Xu tables for horizontal maneuvers are much larger. Recent work explored a new algorithm that exploits the score table's natural symmetry to remove redundancy within the table [9]. However, results showed that this compression algorithm could not achieve sufficient reduction in storage before compromising performance.

WIRED

In 2019, predictive algorithms will start to make banking fair for all

AUTOMATED BACKGROUND CHECKS ARE DECIDING WHO'S FIT FOR A HOME

By Colin Lecher | @colinlecher | Feb 1, 2019, 8:00am EST

WIRED

Machine Bias

There's software used across the country to predict future criminals. And it's biased against blacks.

by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica

May 23, 2016

small-claims cases, part of a push to make government services smarter.

STAT+

IBM's Watson supercomputer recommended 'unsafe and incorrect' cancer treatments, internal documents show

By Casey Ross *and* Ike Swetlitz

July 25, 2018

The Telegraph

BUSINESS NEWS OCTOBER 10, 2018 / 5:12 AM / A YEAR AGO

Amazon scraps secret AI recruiting tool that showed bias against women

Jeffrey Dastin

3,658,826 views | Feb 16, 2012, 11:02am

How Target Figured Out A Teen Girl Was Pregnant Before Her Father Did

Kashmir Hill Former Staff

Tech

Welcome to The Not-So Private Parts where technology & privacy collide

WIRED

In 2019, p… make ban…

AUTO… DECID…

By Colin Lecher | @colinlecher | Feb 1, 2019, 8:00am EST

Millions of black people affected by racial bias in health-care algorithms

Study reveals rampant racism in decision-making software used by US hospitals — and highlights ways to correct it.

24 October 2019

NEWS

Heidi Ledford

A self-driving Uber ran a red light last December, contrary to company claims

Internal documents reveal that the car was at fault

By Andrew Liptak | @AndrewLiptak | Feb 25, 2017, 11:08am EST

Data Science Pipeline

pre-processing

training

data analysis

MACHINE LEARNING

DATA SCIENCE

Data is Dirty

- **inconsistent** data
- **incorrect** data
- **incomplete** data
- **inaccurate** data

Pre-Processing is Fragile

pre-processing

mislabeled data

accidentally duplicated data

wrongly converted data

accidentally (un)used data

TECHNOLOGY

The New York Times

For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights

By Steve Lohr

Aug. 17, 2014

Technology revolutions come in measured, sometimes foot-dragging steps. The lab science and marketing enthusiasm tend to underestimate the bottlenecks to progress that must be overcome with hard work and practical engineering.

The field known as "big data" offers a contemporary case study. The catchphrase stands for the modern abundance of digital data from many sources — the web, sensors, smartphones and corporate databases — that can be mined with clever software for discoveries and insights. Its promise is smarter, data-driven decision-making in every field. That is why data scientist is the economy's hot new job.

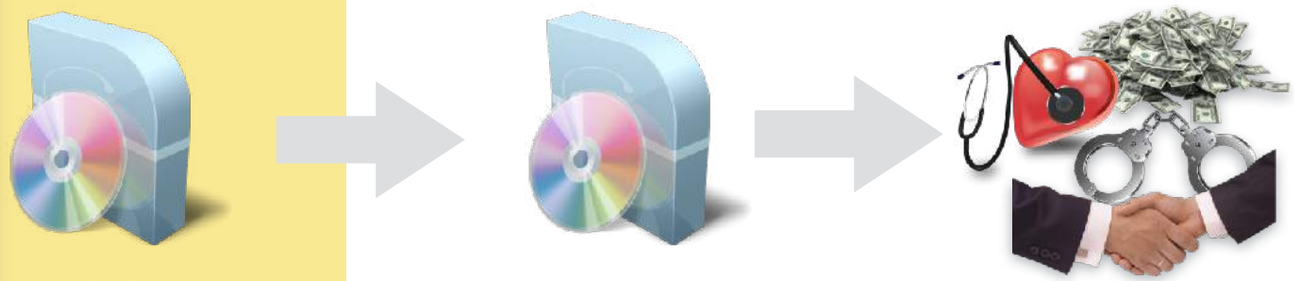
Yet far too much handcrafted work — what data scientists call "data wrangling," "data munging" and "data janitor work" — is still required. Data scientists, according to interviews and expert estimates, spend from 50 percent to 80 percent of their time mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets.

"Data wrangling is a huge — and surprisingly so — part of the job," said Monica Rogati, vice president for data science at Jawbone, whose sensor-filled wristband and software track activity, sleep and food consumption, and suggest dietary and health tips based on the numbers. "It's something that is not appreciated by data civilians. At times, it feels like everything we do."

Several start-ups are trying to break through these big data bottlenecks by developing software to automate the gathering, cleaning and organizing of disparate data, which is plentiful but messy. The modern Wild West of data needs to be tamed somewhat so it can be recognized and exploited by a computer program.

"It's an absolute myth that you can send an algorithm over raw data and have insights pop up," said Jeffrey Heer, a professor of computer science at the University of Washington and a co-founder of Trifacta, a start-up based in San Francisco.

Accuracy is Meaningless



training **data analysis**

Inscrutability

pre-processing training data analysis

MIT Technology Review

Artificial Intelligence / Machine Learning

The Dark Secret at the Heart of AI

No one really knows how the most advanced algorithms do what they do. That could be a problem.

by **Will Knight** Apr 11, 2017

Software = Problems

pre-processing

training

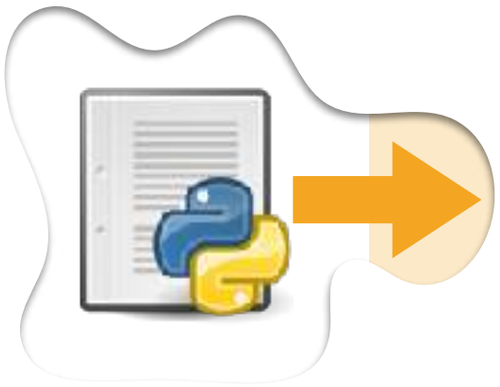
data analysis

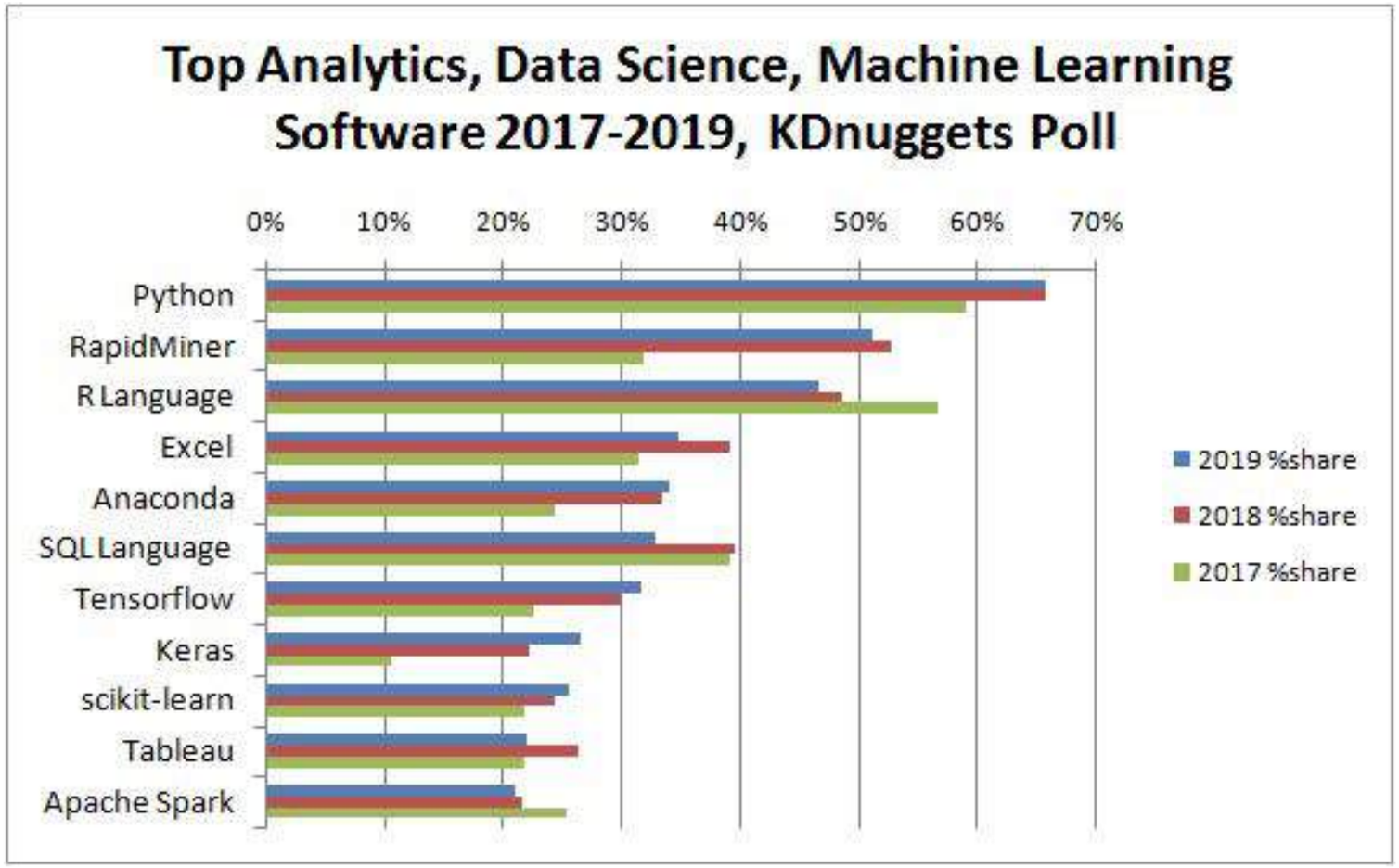
The Lyra Static Analyzer

The Lyra Static Analyzer

Python

Most Popular Programming Language for Data Science

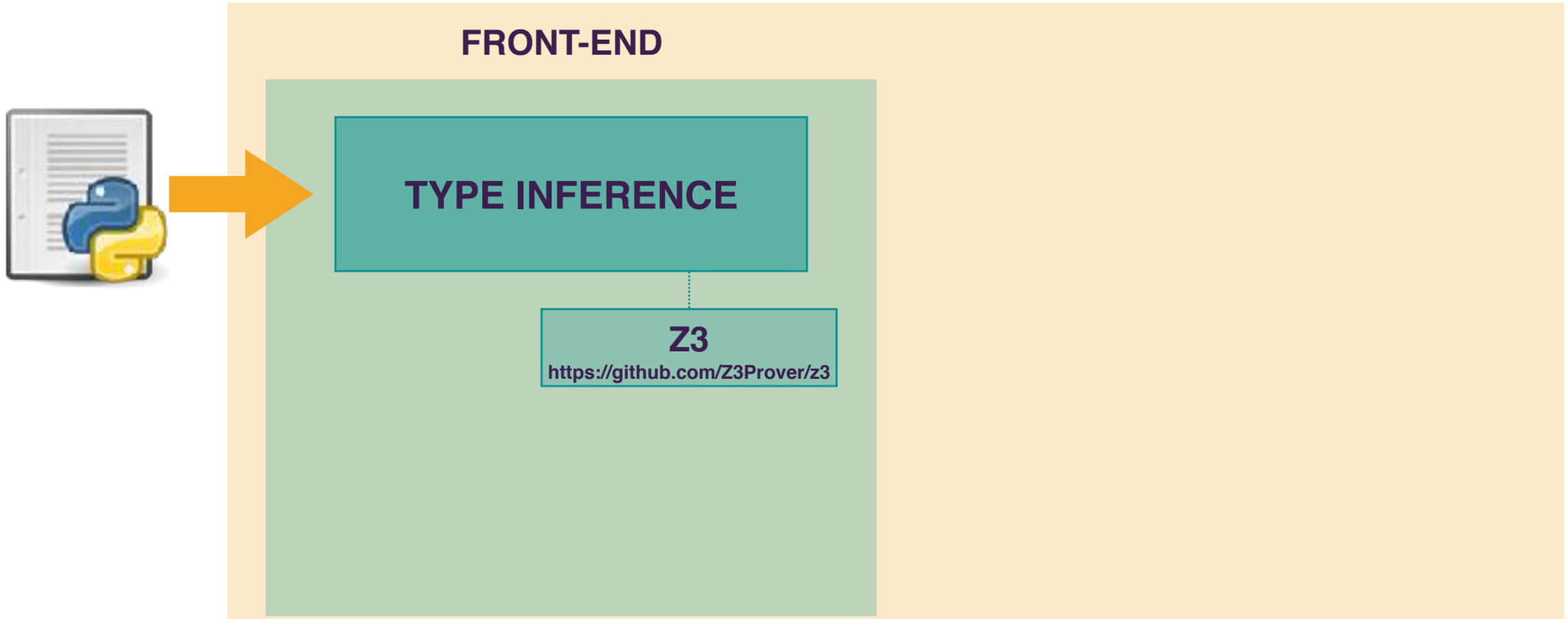




20th annual KDnuggets Software Poll

https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html

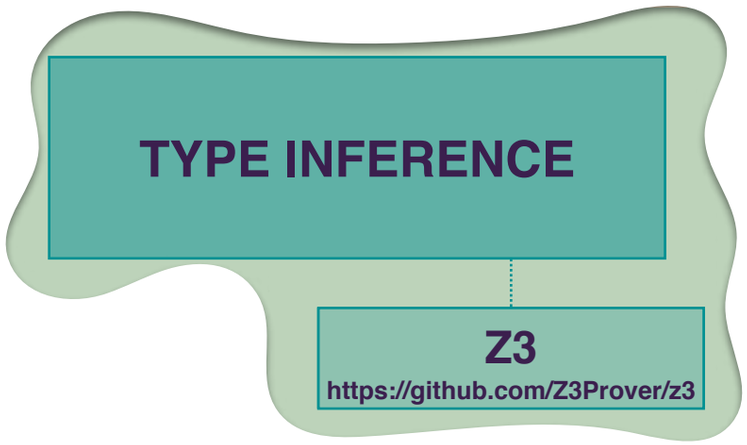
The Lyra Static Analyzer

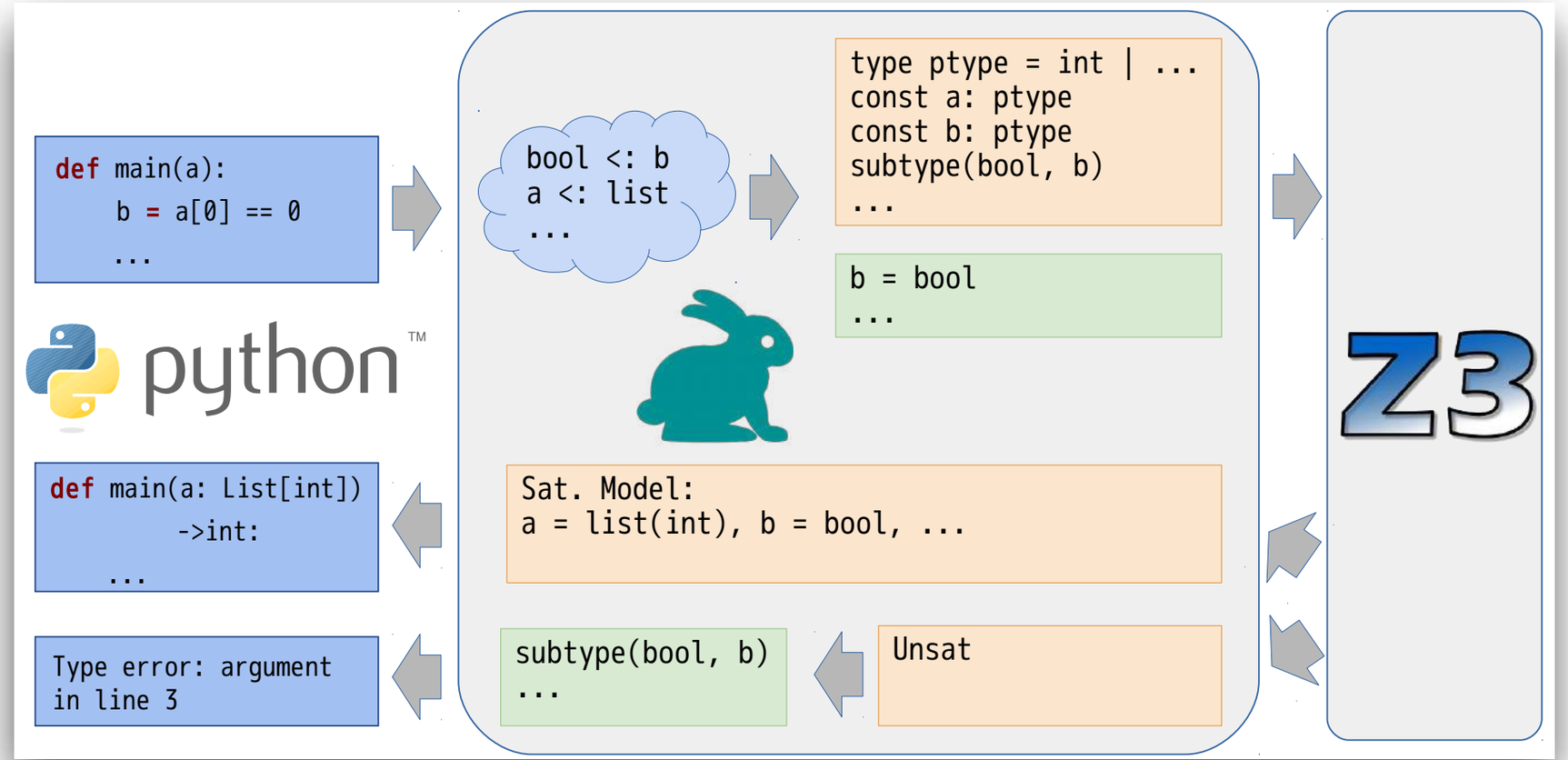


Python is a **dynamically-typed language**

Typpete

SMT-based Static Type Inference for Python 3.x

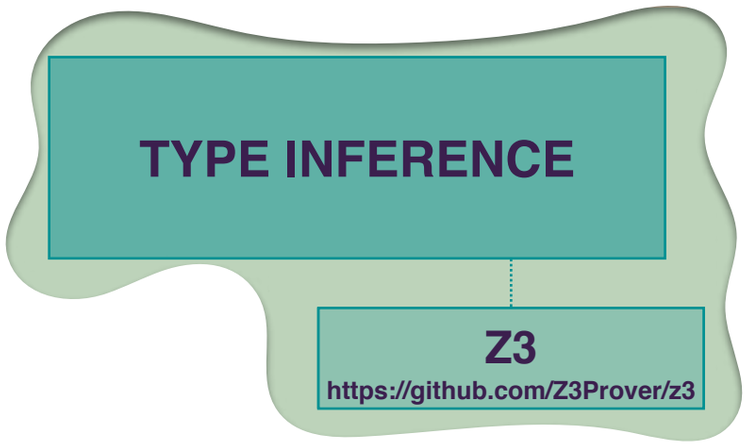


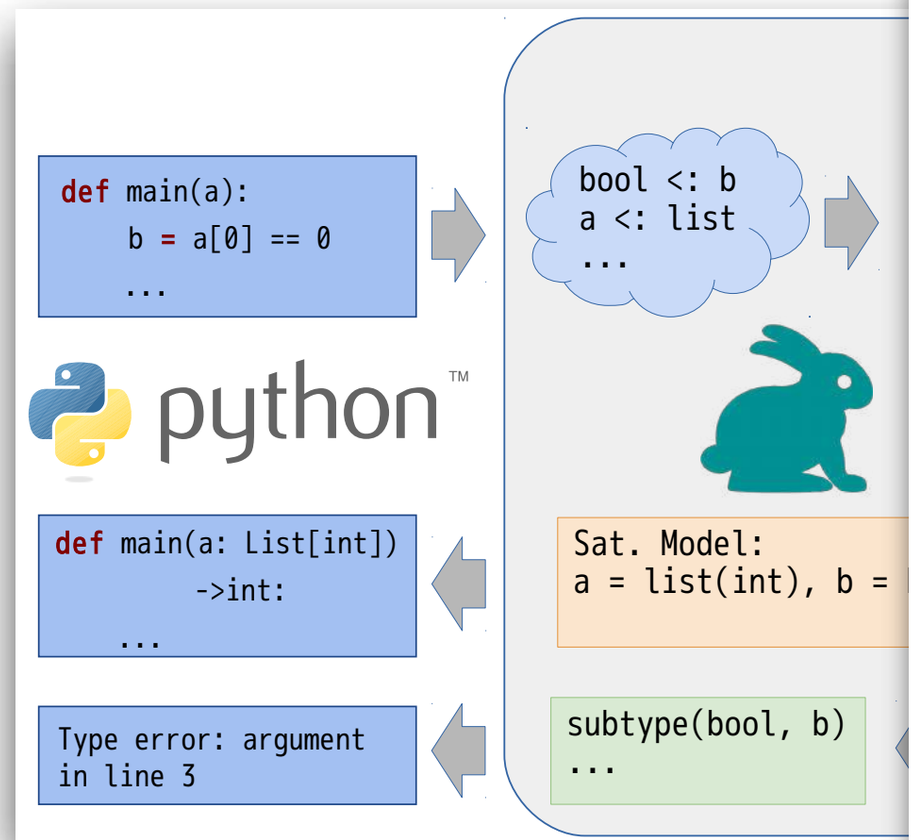


https://github.com/caterinaurban/Typpete

Typpete

SMT-based Static Type Inference for Python 3.x





https://github.com/caterinaurban/Typpete

MaxSMT-Based Type Inference for Python 3

Mostafa Hassan[1,2], Caterina Urban[2(✉)], Marco Eilers[2], and Peter Müller[2]

[1] German University in Cairo, Cairo, Egypt
[2] Department of Computer Science, ETH Zurich, Zurich, Switzerland
caterina.urban@inf.ethz.ch

Abstract. We present Typpete, a sound type inferencer that automatically infers Python 3 type annotations. Typpete encodes type constraints as a MaxSMT problem and uses optional constraints and specific quantifier instantiation patterns to make the constraint solving process efficient. Our experimental evaluation shows that Typpete scales to real world Python programs and outperforms state-of-the-art tools.

1 Introduction

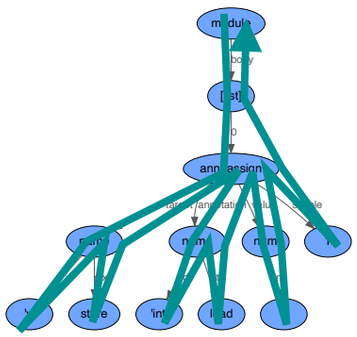
Dynamically-typed languages like Python have become increasingly popular in the past five years. Dynamic typing enables rapid development and adaptation to changing requirements. On the other hand, static typing offers early error detection, efficient execution, and machine-checked code documentation, and enables more advanced static analysis and verification approaches [15].

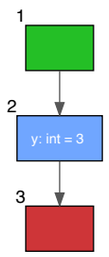
For these reasons, Python's PEP484 [25] has recently introduced optional type annotations in the spirit of gradual typing [23]. The annotations can be checked using MyPy [10]. In this paper, we present our tool Typpete, which automatically infers sound (non-gradual) type annotations and can therefore serve as a preprocessor for other analysis or verification tools.

Typpete performs whole-program type inference, as there are no principal typings in object-oriented languages like Python [1, example in Sect. 1]; the inferred types are correct in the given context but may not be as general as possible. The type inference is constraint-based and relies on the off-the-shelf SMT solver Z3 [7] for finding a valid type assignment for the input program. We show that two main ingredients allow Typpete to scale to real programs: (1) a careful encoding of subtyping that leverages efficient quantifier instantiation techniques [6], and (2) the use of optional type equality constraints, which considerably reduce the solution search space. Whenever a valid type assignment for the input program cannot be found, Typpete encodes type error localization as an optimization problem [19] and reports only a minimal set of unfulfilled constraints to help the user pinpoint the cause of the error.

H. Chockler and G. Weissenbacher (Eds.): CAV 2018, LNCS 10982, pp. 12–19, 2018.
https://doi.org/10.1007/978-3-319-96142-2_2

The Lyra Static Analyzer

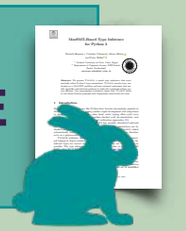




FRONT-END

https://github.com/caterinaurban/Typpete

TYPE INFERENCE

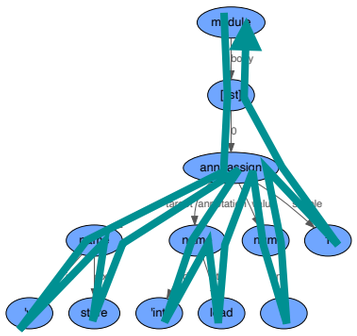


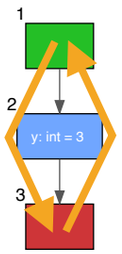
Z3

https://github.com/Z3Prover/z3

CFG GENERATOR

The Lyra Static Analyzer





FRONT-END

ABSTRACT DOMAINS

https://github.com/caterinaurban/Typpete

TYPE INFERENCE

Z3

https://github.com/Z3Prover/z3

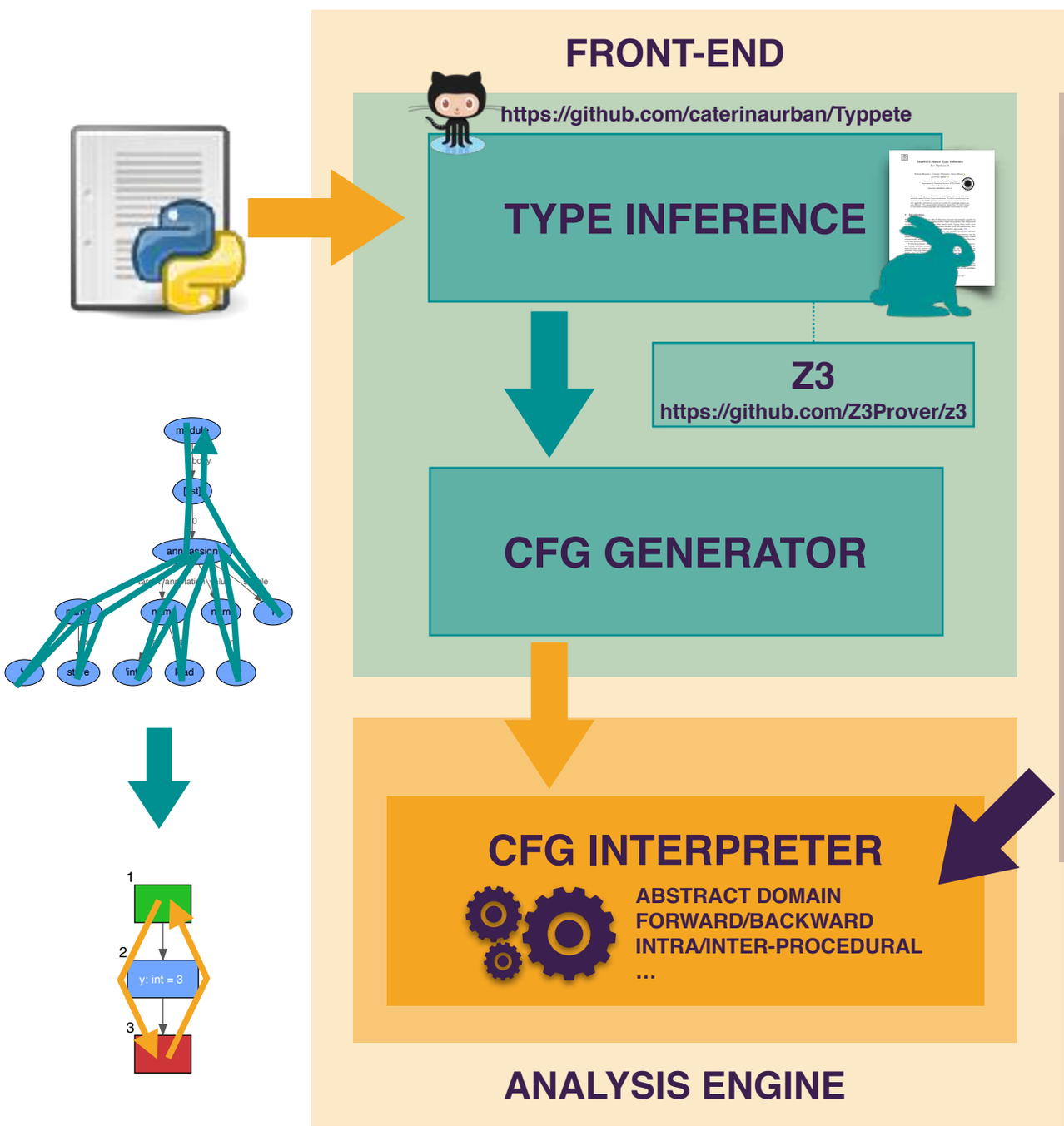
CFG GENERATOR

CFG INTERPRETER

ABSTRACT DOMAIN
FORWARD/BACKWARD
INTRA/INTER-PROCEDURAL
...

ANALYSIS ENGINE

The Lyra Static Analyzer



Check for updates

Evaluating Design Tradeoffs in Numeric Static Analysis for Java

Shiyi Wei[1(✉)], Piotr Mardziel[2], Andrew Ruef[3], Jeffrey S. Foster[3], and Michael Hicks[3]

[1] The University of Texas at Dallas, Richardson, USA
swei@utdallas.edu
[2] Carnegie Mellon University, Moffett Field, USA
piotrm@gmail.com
[3] University of Maryland, College Park, USA
{awruef,jfoster,mwh}@cs.umd.edu

Abstract. Numeric static analysis for Java has a broad range of potentially useful applications, including array bounds checking and resource usage estimation. However, designing a scalable numeric static analysis for real-world Java programs presents a multitude of design choices, each of which may interact with others. For example, an analysis could handle method calls via either a top-down or bottom-up interprocedural analysis. Moreover, this choice could interact with how we choose to represent aliasing in the heap and/or whether we use a relational numeric domain, e.g., convex polyhedra. In this paper, we present a family of abstract interpretation-based numeric static analyses for Java and systematically evaluate the impact of 162 analysis configurations on the DaCapo benchmark suite. Our experiment considered the precision and performance of the analyses for discharging array bounds checks. We found that top-down analysis is generally a better choice than bottom-up analysis, and that using access paths to describe heap objects is better than using summary objects corresponding to points-to analysis locations. Moreover, these two choices are the most significant, while choices about the numeric domain, representation of abstract objects, and context-sensitivity make much less difference to the precision/performance tradeoff.

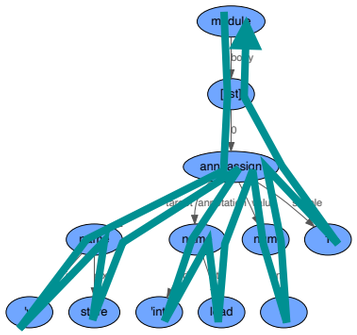
1 Introduction

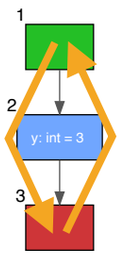
Static analysis of numeric program properties has a broad range of useful applications. Such analyses can potentially detect array bounds errors [50], analyze a program's resource usage [28,30], detect side channels [8,11], and discover vectors for denial of service attacks [10,26].

One of the major approaches to numeric static analysis is abstract interpretation [18], in which program statements are evaluated over an abstract domain until a fixed point is reached. Indeed, the first paper on abstract interpretation [18] used numeric intervals as one example abstract domain,

A. Ahmed (Ed.): ESOP 2018, LNCS 10801, pp. 653–682, 2018.
https://doi.org/10.1007/978-3-319-89884-1_23

The Lyra Static Analyzer





FRONT-END

https://github.com/caterinaurban/Typpete

TYPE INFERENCE

Z3

https://github.com/Z3Prover/z3

CFG GENERATOR

CFG INTERPRETER

ABSTRACT DOMAIN
FORWARD/BACKWARD
INTRA/INTER-PROCEDURAL
...

ANALYSIS ENGINE

ABSTRACT DOMAINS

NUMERICAL

signs

...

boxes

octagons

polyhedra

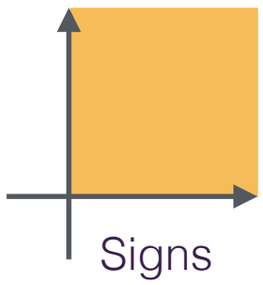
APRONPY

APRON

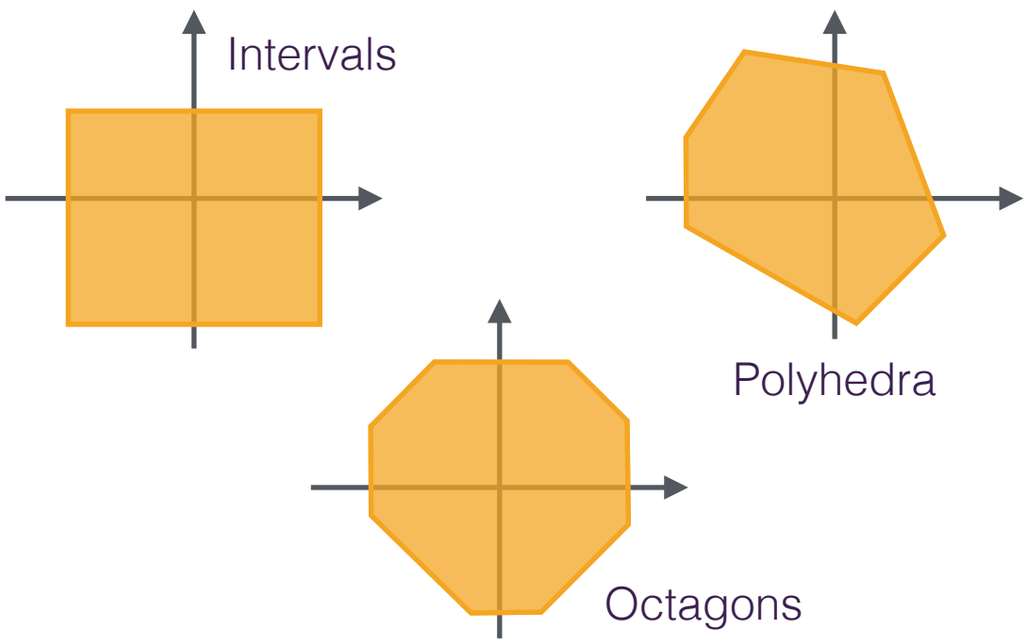
https://github.com/antoinemine/apron

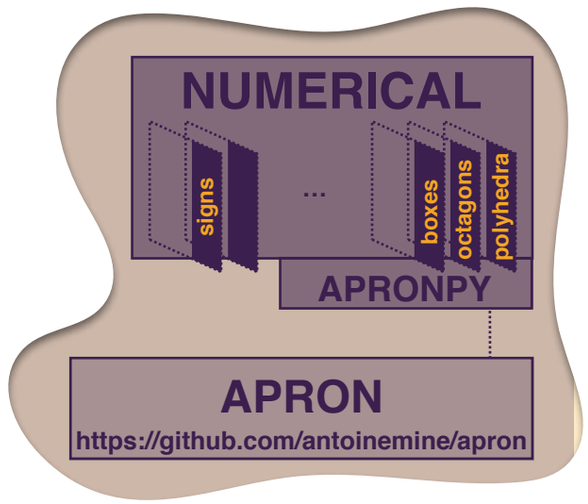
Numerical Abstract Domains

Implemented Natively



Built on APRON



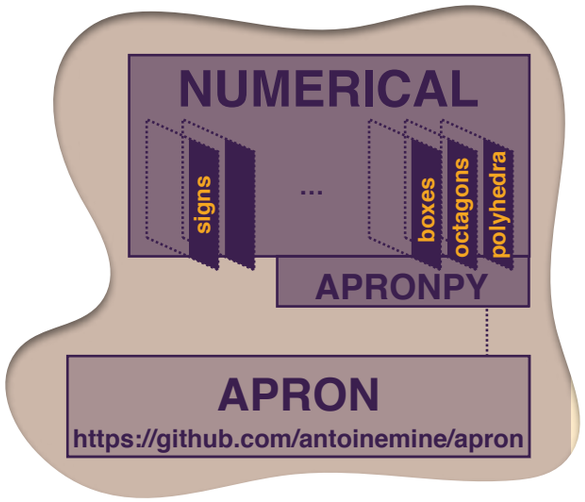


ApronPy

Python Bindings for the APRON Numerical Abstract Domain Library

```c
typedef struct ap_abstract1_t {
  ap_abstract0_t* abstract0;
  ap_environment_t* env;
} ap_abstract1_t;

...

bool ap_abstract1_is_bottom(ap_manager_t* man, ap_abstract1_t* a);
bool ap_abstract1_is_top(ap_manager_t* man, ap_abstract1_t* a);
bool ap_abstract1_is_leq(ap_manager_t* man, ap_abstract1_t* a1, ap_abstract1_t* a2);
bool ap_abstract1_is_eq(ap_manager_t* man, ap_abstract1_t* a1, ap_abstract1_t* a2);
```

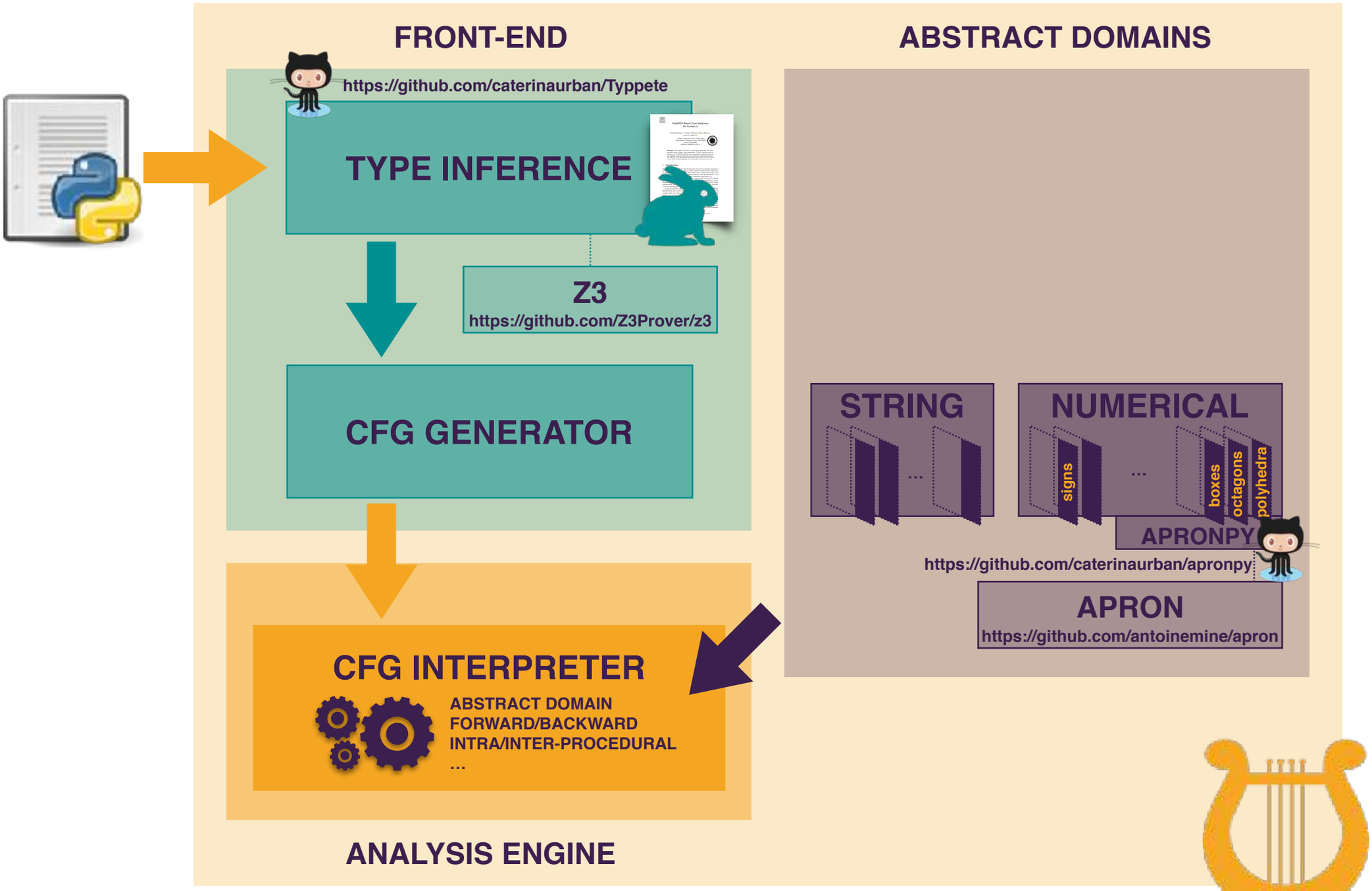
```python
class Abstract1(Structure):
    _fields_ = [
        ('abstract0', POINTER(Abstract0)),
        ('env', POINTER(Environment))
    ]

class PyAbstract1(metaclass=ABCMeta):
    ...

    def is_bottom(self):
        return bool(libapron.ap_abstract1_is_bottom(self.manager, self))

    def is_top(self):
        return bool(libapron.ap_abstract1_is_top(self.manager, self))

    def is_leq(self, other: 'PyAbstract1'):
        return bool(libapron.ap_abstract1_is_leq(self.manager, self, other))

    def is_eq(self, other: 'PyAbstract1'):
        return bool(libapron.ap_abstract1_is_eq(self.manager, self, other))
```



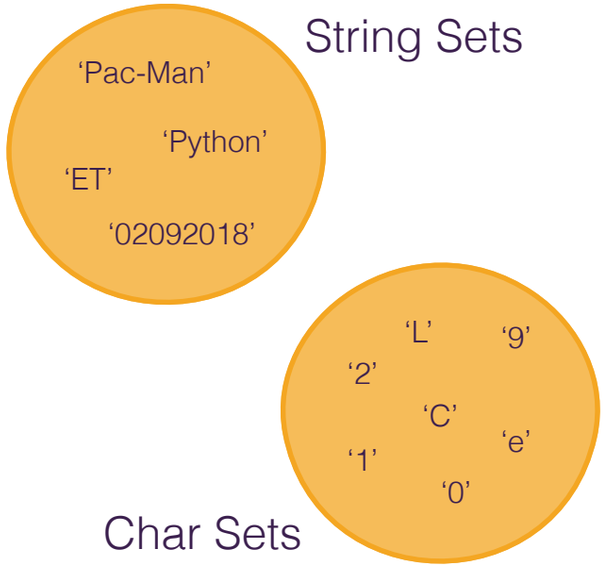
ctypes

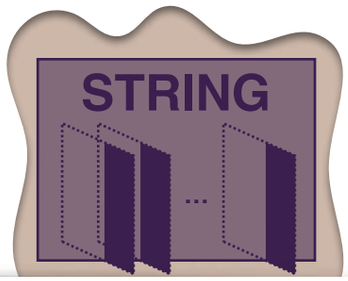
https://github.com/caterinaurban/apronpy

The Lyra Static Analyzer



String Abstract Domains





SOFTWARE – PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2015; **45**:245–287
Published online 16 August 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.2218

A suite of abstract domains for static analysis of string values

Giulia Costantini[1,*,†], Pietro Ferrara[2,3] and Agostino Cortesi[1]

[1]*Ca' Foscari University, Mestre, Venezia, Italy*
[2]*ETH, Zurich, Switzerland*
[3]*IBM Thomas J. Watson Research Center, USA*

SUMMARY

Strings are widely used in modern programming languages in various scenarios. For instance, strings are used to build up Structured Query Language (SQL) queries that are then executed. Malformed strings may lead to subtle bugs, as well as non-sanitized strings may raise security issues in an application. For these reasons, the application of static analysis to compute safety properties over string values at compile time is particularly appealing. In this article, we propose a generic approach for the static analysis of string values based on abstract interpretation. In particular, we design a suite of abstract semantics for strings, where each abstract domain tracks a different kind of information. We discuss the trade-off between efficiency and accuracy when using such domains to catch the properties of interest. In this way, the analysis can be tuned at different levels of precision and efficiency, and it can address specific properties. Copyright © 2013 John Wiley & Sons, Ltd.

Received 23 November 2012; Revised 8 May 2013; Accepted 16 July 2013

KEY WORDS: static analysis; abstract interpretation; abstract domains; strings

Strings are widely used in modern prog
an output to a user to the constructio
Java, they are widely used to build u
about the classes through reflection. Th
For instance, the execution of `str.s`
`str` does not contain an `'a'` charac
characters surely contained on the vari
we execute the query 'DELETE FRO
OR TRUE'? The content of `Table` w
lation of strings could lead not only to
too [1].

The interest on approaches that autc
raising. The state of the art in this field
ular expressions are precise but slow, a
are focused on particular properties
ity are the main advantages of the ab
define analyses at different levels of pr
interpretation as an alternative approac

*Correspondence to: Giulia Costantini, Com
Venice, Italy.

†E-mail: costantini@dsi.unive.it

Static Program Analysis for String Manipulation Languages

Vincenzo Arceri
University of Verona, Verona, Italy
vincenzo.arceri@univr.it

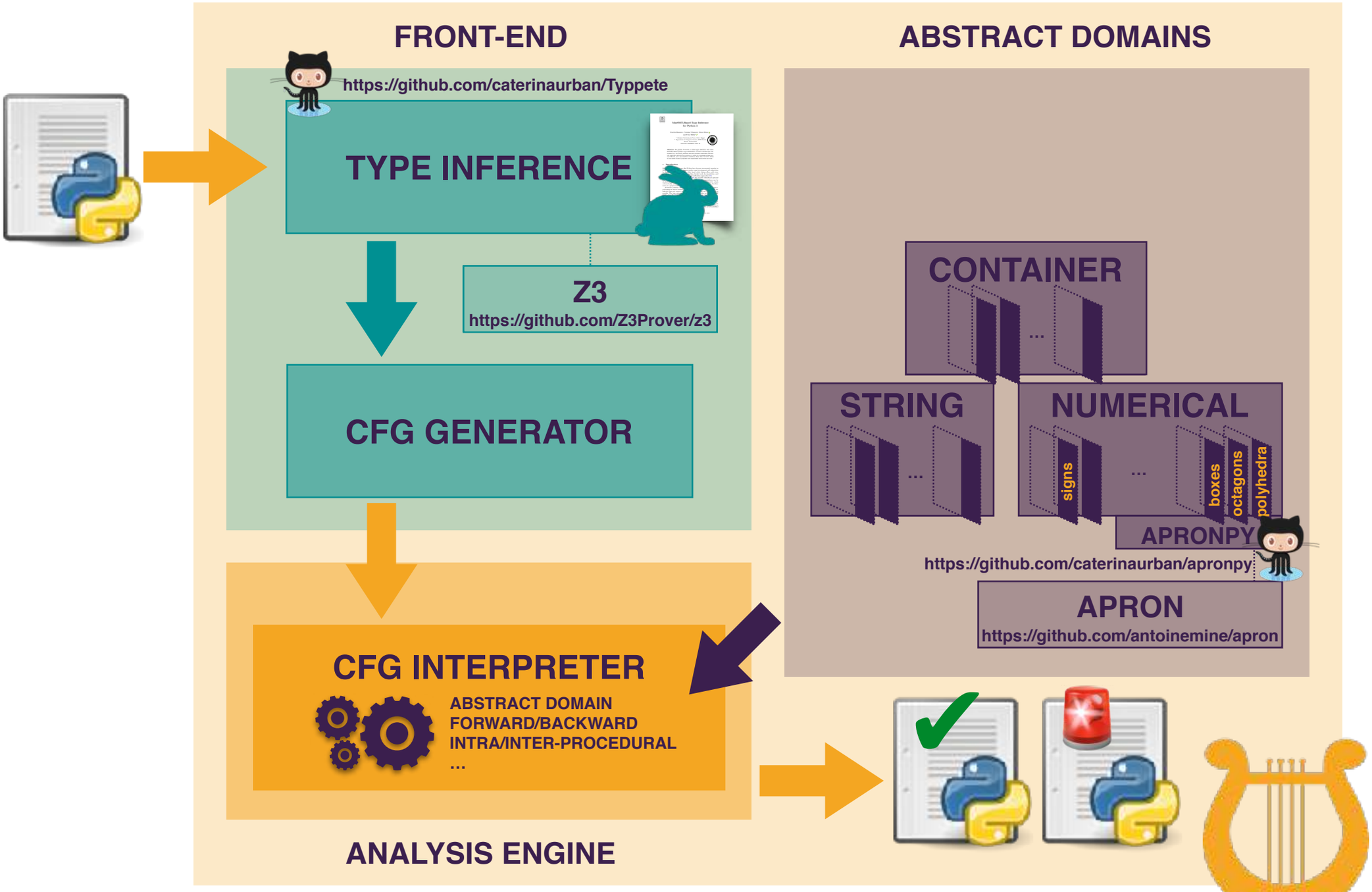
Isabella Mastroeni
University of Verona, Verona, Italy
isabella.mastroeni@univr.it

In recent years, dynamic languages, such as JavaScript or Python, have been increasingly used in a wide range of fields and applications. Their tricky and misunderstood behaviors pose a hard challenge for static analysis of these programming languages. A key aspect of any dynamic language program is the multiple usage of strings, since they can be implicitly converted to another type value, transformed by string-to-code primitives or used to access an object-property. Unfortunately, string analyses for dynamic languages still lack precision and do not take into account some important string features. Moreover, string obfuscation is very popular in the context of dynamic language malicious code, for example, to hide code information inside strings and then to dynamically transform strings into executable code. In this scenario, more precise string analyses become a necessity. This paper is placed in the context of static string analysis by abstract interpretation and proposes a new semantics for string analysis, placing a first step for handling dynamic languages string features.

1 Introduction

Dynamic languages, such as JavaScript or Python, have faced an important increment of usage in a very wide range of fields and applications. Common features in dynamic languages are dynamic typing (typing occurs during program execution, at run-time) and implicit type conversion [38], lightening the development phase and allowing not to block the program execution in presence of unexpected or unpre-

The Lyra Static Analyzer



Container Abstract Domains

Default: Expansion + Summarization

Numeric Domains with Summarized Dimensions

Denis Gopan[1], Frank DiMaio[1], Nurit Dor[2], Thomas Reps[1], and Mooly Sagiv[2]

[1] Comp. Sci. Dept., University of Wisconsin; {gopan,dimaio,reps}@cs.wisc.edu
[2] School of Comp. Sci., Tel-Aviv University; {nurr,msagiv}@post.tau.ac.il

Abstract. We introduce a systematic approach to designing *summarizing abstract numeric domains* from existing numeric domains. Summarizing domains use *summary* dimensions to represent potentially unbounded collections of numeric objects. Such domains are of benefit to analyses that verify properties of systems with an unbounded number of numeric objects, such as shape analysis, or systems in which the number of numeric objects is bounded, but large.

1 Introduction

Verifying the correctness of complex software systems requires reasoning about numeric quantities. In particular, an analysis technique may have to discover certain relationships among values of *numeric objects*, such as numeric variables, numeric array elements, or numeric-valued fields of heap-allocated structures [2]. For example, to verify that there are no buffer overruns in a particular C program, an analysis needs to make sure that the value of an index variable does not exceed the length of the buffer at each program point where the buffer is accessed [16].

Numeric analyses have been a research topic for several decades, and a number of numeric domains that allow to approximate numeric state of a system have been

A Parametric Segmentation Functor for Fully Automatic and Scalable Array Content Analysis

Patrick Cousot
École normale supérieure &
New York University
Courant Institute of Mathematical Sciences
cousot@ens.fr, pcousot@cs.nyu.edu

Radhia Cousot
Centre National de la Recherche Scientifique
École normale supérieure &
Microsoft Research, Redmond
radhia.cousot@ens.fr

Francesco Logozzo
Microsoft Research, Redmond
logozzo@microsoft.com

Abstract

We introduce Fun
main functor for th
content properties.
cient lifting of exis
analysis of uniform
collections. The a
arrays into consecu
Segments are delin
uniformly. All sym
equal in the concr
via reduced produc
The analysis is pres
the choices of boun
duction operator. C
parameters, the ana
We first prototy
iment with the abst
priate precision/rat
an abstract interpr
We empirically val
analysis by running
code. We were able
verify the impleme
the best of our kno
plied to such a larg

Categories and Su

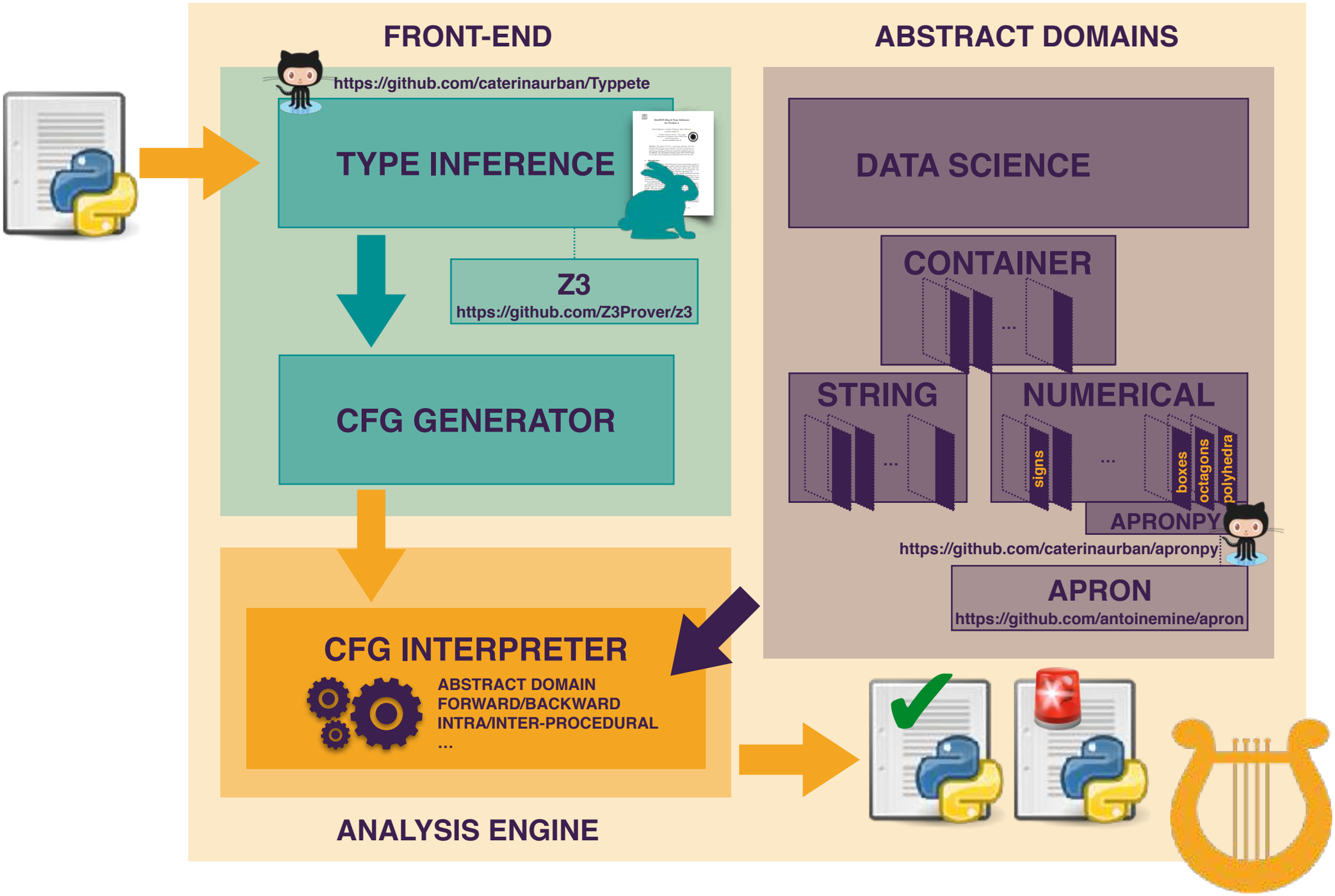
Available online at www.sciencedirect.com
SciVerse ScienceDirect
Electronic Notes in Theoretical Computer Science
Electronic Notes in Theoretical Computer Science 287 (2012) 53–64
www.elsevier.com/locate/entcs

Generic Abstraction of Dictionaries and Arrays

Jędrzej Fulara[1,2]

Institute of Informatics
University of Warsaw

The Lyra Static Analyzer



Data Usage Abstract Domains

pre-processing **training** **data analysis**

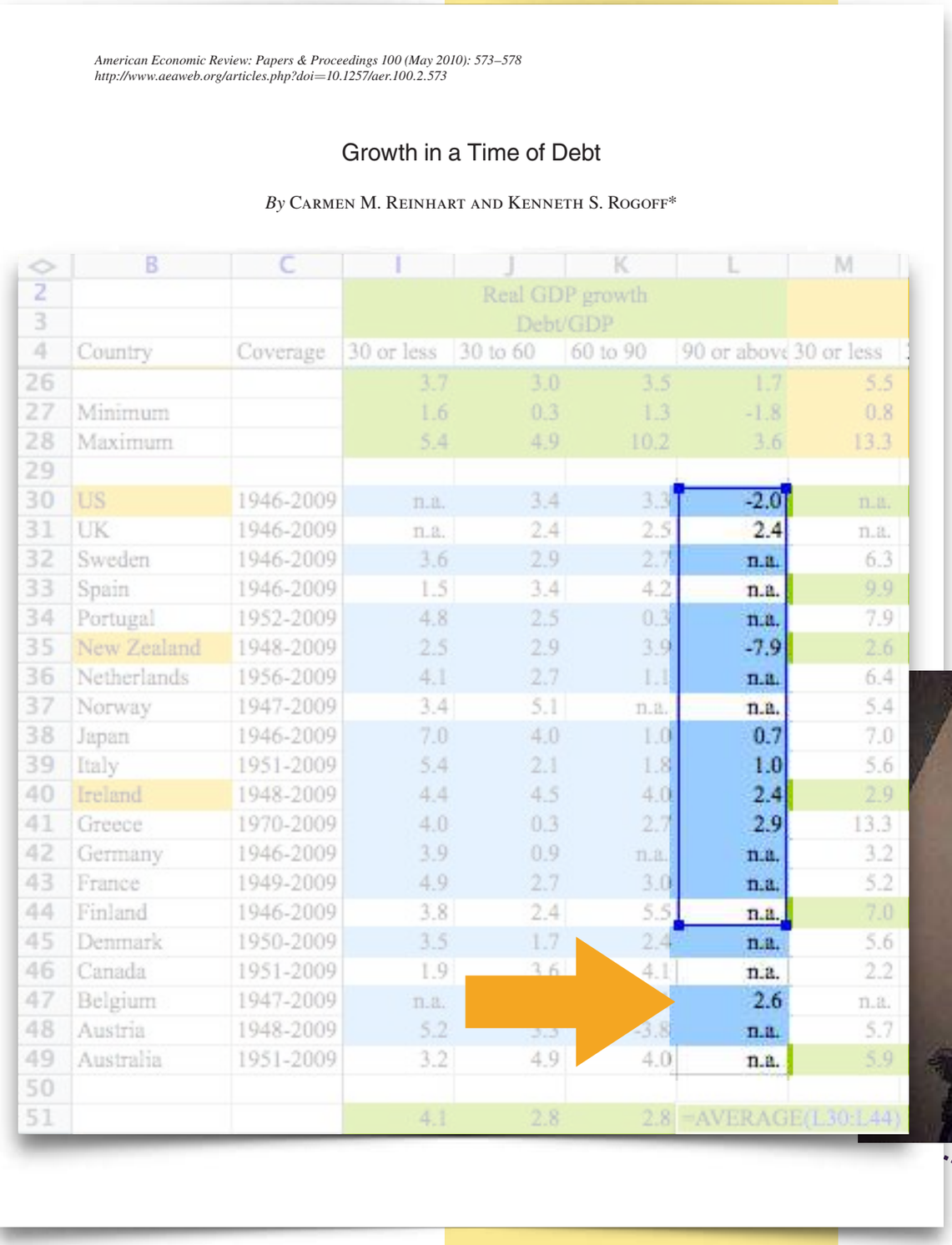
mislabeled data

accidentally duplicated data

wrongly converted data

accidentally (un)used data

Data Usage Abstract Domains

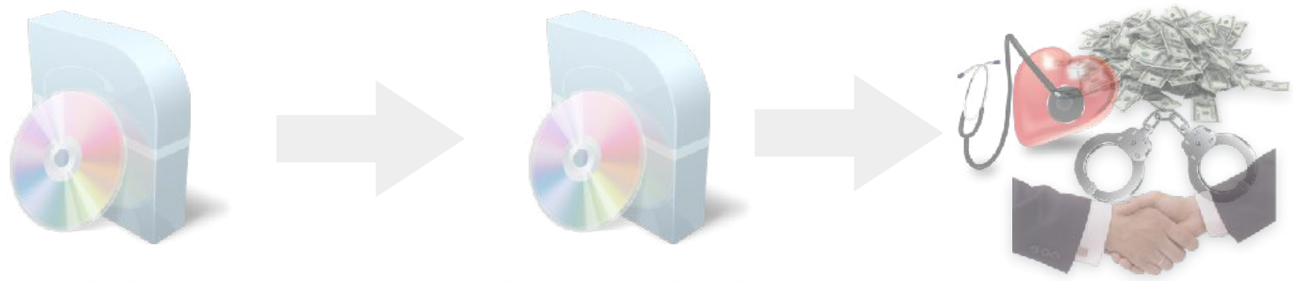


American Economic Review: Papers & Proceedings 100 (May 2010): 573–578
http://www.aeaweb.org/articles.php?doi=10.1257/aer.100.2.573

Growth in a Time of Debt

By Carmen M. Reinhart and Kenneth S. Rogoff*

	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.			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)	



training data analysis

accidentally (un)used data

Data Usage Abstract Domains

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*

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

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?

FACEBOOK
TWITTER
GOOGLE+
SAVE
EMAIL
SHARE
PRINT
REPRINTS

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

Data Usage Abstract Domains

Check for updates

An Abstract Interpretation Framework for Input Data Usage

Caterina Urban(✉) and Peter Müller

Department of Computer Science, ETH Zurich, Zurich, Switzerland
{caterina.urban,peter.mueller}@inf.ethz.ch

Abstract. Data science software plays an increasingly important role in critical decision making in fields ranging from economy and finance to biology and medicine. As a result, errors in data science applications can have severe consequences, especially when they lead to results that look plausible, but are incorrect. A common cause of such errors is when applications erroneously ignore some of their input data, for instance due to bugs in the code that reads, filters, or clusters it.

In this paper, we propose an abstract interpretation framework to automatically detect unused input data. We derive a program semantics that precisely captures data usage by abstraction of the program's operational trace semantics and express it in a constructive fixpoint form. Based on this semantics, we systematically derive static analyses that automatically detect unused input data by fixpoint approximation.

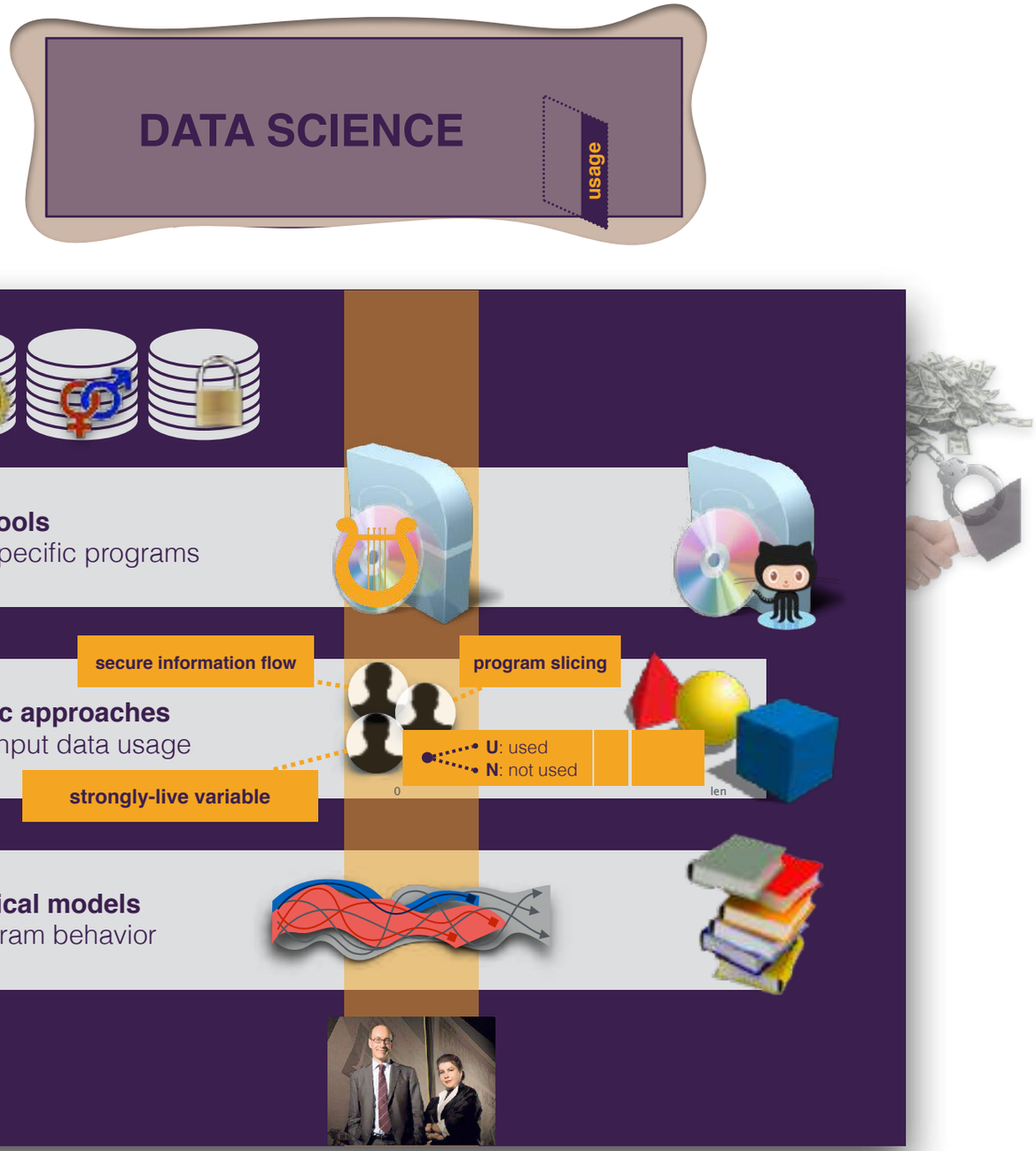
This clear design principle provides a framework that subsumes existing analyses; we show that secure information flow analyses and a form of live variables analysis can be used for data usage, with varying degrees of precision. Additionally, we derive a static analysis to detect single unused data inputs, which is similar to dependency analyses used in the context of backward program slicing. Finally, we demonstrate the value of expressing such analyses as abstract interpretation by combining them with an existing abstraction of compound data structures such as arrays and lists to detect unused chunks of the data.

1 Introduction

In the past few years, data science has grown considerably in importance and now heavily influences many domains, ranging from economy and finance to biology and medicine. As we rely more and more on data science for making decisions, we become increasingly vulnerable to programming errors.

Programming errors can cause frustration, especially when they lead to a program failure after hours of computation. However, programming errors that do not cause failures can have more serious consequences as code that produces an erroneous but plausible result gives no indication that something went wrong. A notable example is the paper "Growth in a Time of Debt" published in 2010 by economists Reinhart and Rogoff, which was widely cited in political debates and

A. Ahmed (Ed.): ESOP 2018, LNCS 10801, pp. 683–710, 2018.
https://doi.org/10.1007/978-3-319-89884-1_24



accidentally (un)used data

Data Usage Abstract Domains

An Abstract Interpretation Framework for Input Data Usage

Caterina Urban(✉) and Peter Müller

Department of Computer Science, ETH Zurich, Zurich
{caterina.urban,peter.mueller}@inf.eth

Abstract. Data science software plays an increasingly
in critical decision making in fields ranging from econo
to biology and medicine. As a result, errors in data scie
can have severe consequences, especially when they lead
look plausible, but are incorrect. A common cause of suc
applications erroneously ignore some of their input data,
to bugs in the code that reads, filters, or clusters it.

In this paper, we propose an abstract interpretatio
automatically detect unused input data. We derive a pr
that precisely captures data usage by abstraction of the
ational trace semantics and express it in a constructiv
Based on this semantics, we systematically derive stat
automatically detect unused input data by fixpoint app

This clear design principle provides a framework that
ing analyses; we show that secure information flow analys
live variables analysis can be used for data usage, with
of precision. Additionally, we derive a static analysis
unused data inputs, which is similar to dependency anal
context of backward program slicing. Finally, we demon
of expressing such analyses as abstract interpretation by
with an existing abstraction of compound data structure
and lists to detect unused chunks of the data.

1 Introduction

In the past few years, data science has grown considerably in importance and now heavily influences many domains, ranging from economy and finance to biology and medicine. As we rely more and more on data science for making decisions, we become increasingly vulnerable to programming errors.

Programming errors can cause frustration, especially when they lead to a program failure after hours of computation. However, programming errors that do not cause failures can have more serious consequences as code that produces an erroneous but plausible result gives no indication that something went wrong. A notable example is the paper "Growth in a Time of Debt" published in 2010 by economists Reinhart and Rogoff, which was widely cited in political debates and

A. Ahmed (Ed.): ESOP 2018, LNCS 10801, pp. 683–710, 2018.
https://doi.org/10.1007/978-3-319-89884-1_24

Perfectly Parallel Fairness Certification of Neural Networks

CATERINA URBAN, INRIA and DIENS, École Normale Supérieure, CNRS, PSL University, France
MARIA CHRISTAKIS, MPI-SWS, Germany
VALENTIN WÜSTHOLZ, ConsenSys Diligence, Germany
FUYUAN ZHANG, MPI-SWS, Germany

Recently, there is growing concern that machine-learning models, which currently assist or even automate decision making, reproduce, and in the worst case reinforce, bias of the training data. The development of tools and techniques for certifying fairness of these models or describing their biased behavior is, therefore, critical. In this paper, we propose a *perfectly parallel* static analysis for certifying *causal fairness* of feed-forward neural networks used for classification of tabular data. When certification succeeds, our approach provides definite guarantees, otherwise, it describes and quantifies the biased behavior. We design the analysis to be *sound*, in practice also *exact*, and configurable in terms of scalability and precision, thereby enabling *pay-as-you-go certification*. We implement our approach in an open-source tool and demonstrate its effectiveness on models trained with popular datasets.

1 INTRODUCTION

Due to the tremendous advances in machine learning and the vast amounts of available data, software systems, and neural networks in particular, are of ever-increasing importance in our everyday decisions, whether by assisting them or by autonomously making them. We are already witnessing the wide adoption and societal impact of such software in criminal justice, health care, and social welfare, to name a few examples. It is, therefore, not far-fetched to imagine a future where most of the decision making is automated.

However, several studies have recently raised concerns about the fairness of such systems. For instance, consider a commercial recidivism-risk assessment algorithm that was found racially biased [Larson et al. 2016]. Similarly, a commercial algorithm that is widely used in the U.S. health care system falsely determined that Black patients were healthier than other equally sick patients by using health costs to represent health needs [Obermeyer et al. 2019]. There is also empirical evidence of gender bias in image searches, for instance, there are fewer results depicting women when searching for certain occupations, such as CEO [Kay et al. 2015]. Commercial facial recognition algorithms, which are increasingly used in law enforcement, are less effective for women and darker skin types [Buolamwini and Gebru 2018].

In other words, machine-learning software may reproduce, or even reinforce, bias that is directly or indirectly present in the training data. This awareness will certainly lead to regulations and strict audits in the future. It is, therefore, critical to develop tools and techniques for certifying fairness of neural networks and understanding the circumstances of their potentially biased behavior.

Causal Fairness. We make a step forward in meeting these needs by designing a static analysis framework for certifying *causal fairness* [Galhotra et al. 2017] of feed-forward neural networks used for classification tasks. Specifically, given a choice (e.g., driven by a causal model) of input features that are considered (directly or indirectly) sensitive to bias, *a neural network is causally fair if the output classification is not affected by different values of the chosen features.* Note that, unlike

Authors' addresses: Caterina Urban, INRIA , DIENS, École Normale Supérieure, CNRS, PSL University, Paris, France, caterina.urban@inria.fr; Maria Christakis, MPI-SWS, Germany, maria@mpi-sws.org; Valentin Wüstholz, ConsenSys Diligence, Germany, valentin.wustholz@consensys.net; Fuyuan Zhang, MPI-SWS, Germany, fuyuan@mpi-sws.org.

Data Shape Abstract Domains

Implicit Assumptions on the Input Data

pre-processing

training

data analysis

```python
import sys

grade2gpa = { 'A': 4.0, 'B': 3.0, 'C': 2.0, 'D': 1.0, 'F': 0.0 }
with open(sys.argv[1]) as file:
    for line in file:
        data = line.strip().split(' ')
        grades = int(data[1])
        gpa = 0.0
        for i in range(2, grades + 2):
            gpa += grade2gpa[data[i]]
        result = gpa / grades

        print('{}: {}'.format(data[0], result))
```

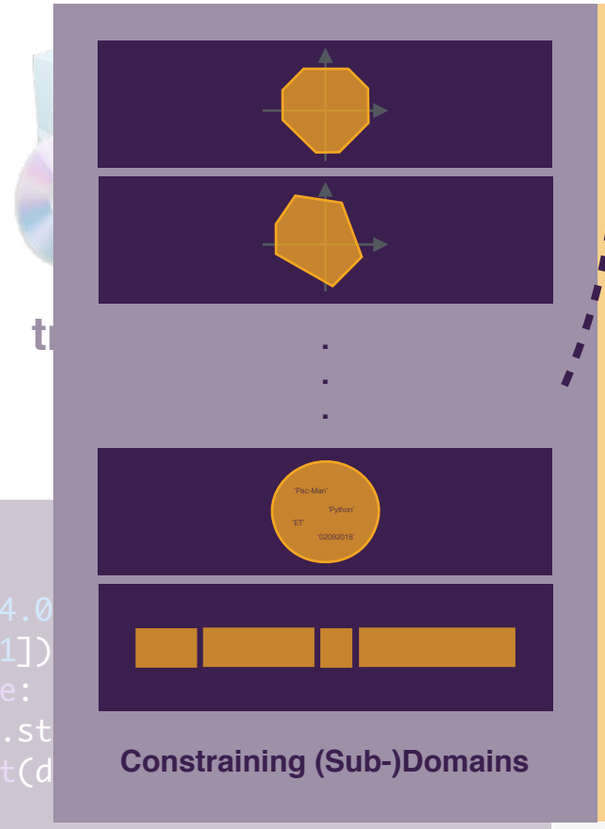
```
dkd 2 A C
ndd 1 F
dle 3 C C C
bk2 1 B+
wwb 2 D F
```

```
dkd 1 A C
ndd 1 F
dle 3 C C C
bk2 1 B
wwb 1 D
```

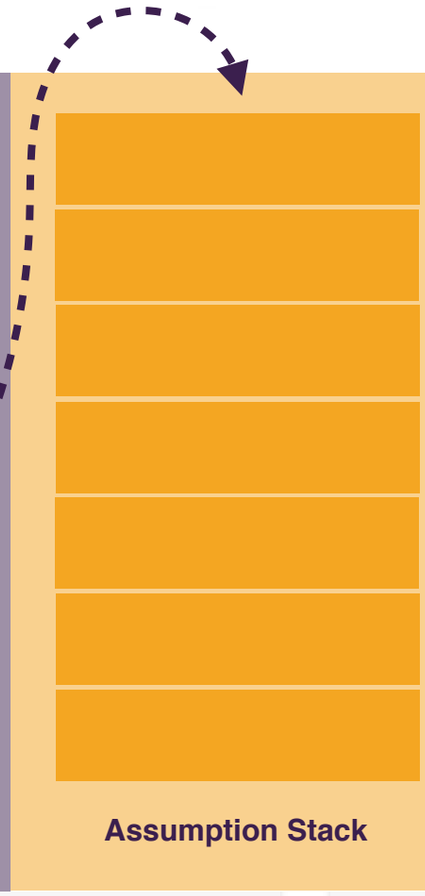
Data Shape Abstract Domains

Implicit Assumptions on the Input Data

pre-processing



Constraining (Sub-)Domains



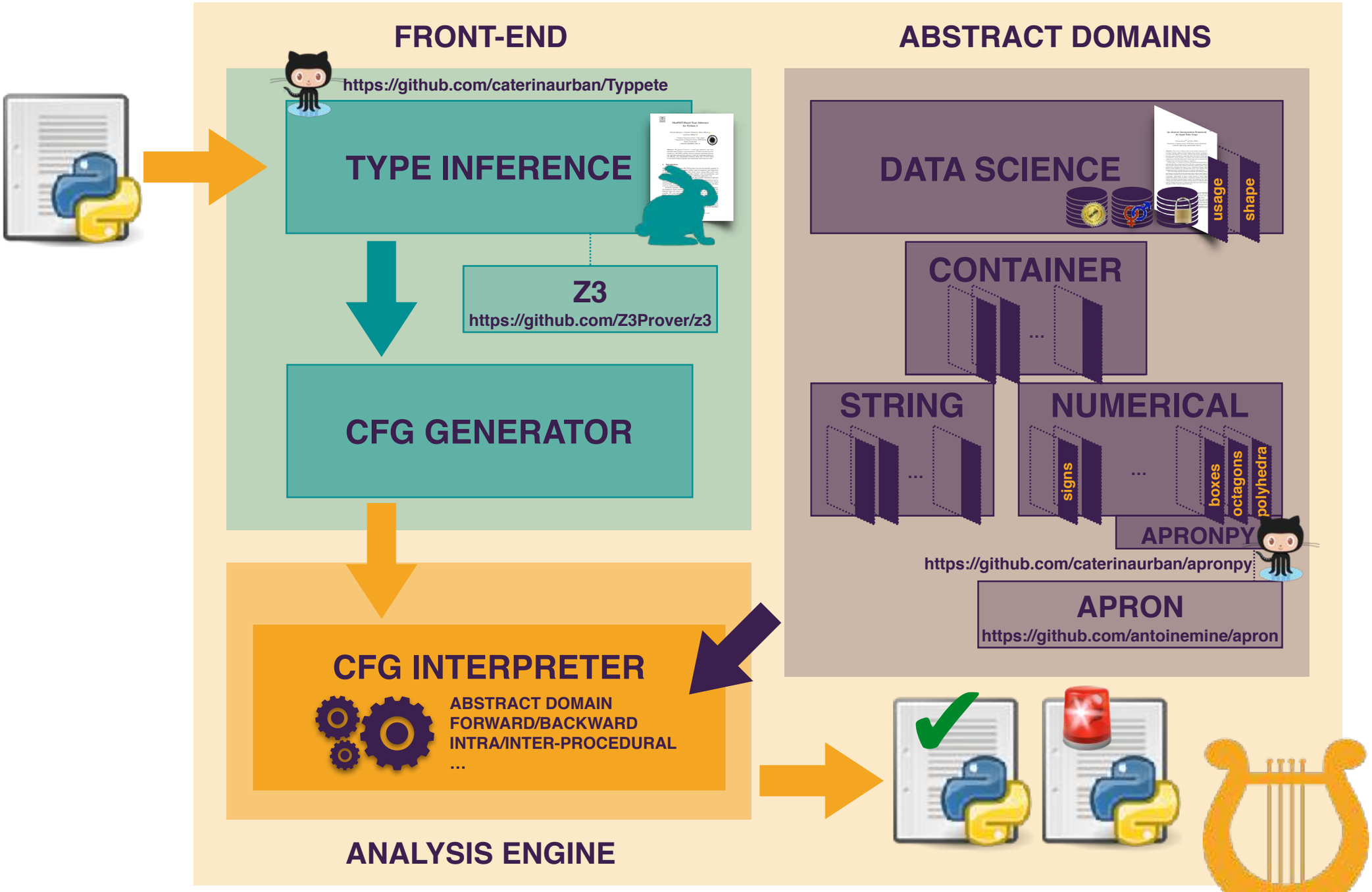
Assumption Stack

```
import sys

grade2gpa = { 'A': 4.0
with open(sys.argv[1])
    for line in file:
        data = line.st
        grades = int(d
        gpa = 0.0
        for i in range(2, grades + 2):
            gpa += grade2gpa[data[i]]
        result = gpa / grades

        print('{}: {}'.format(data[0], result))
```

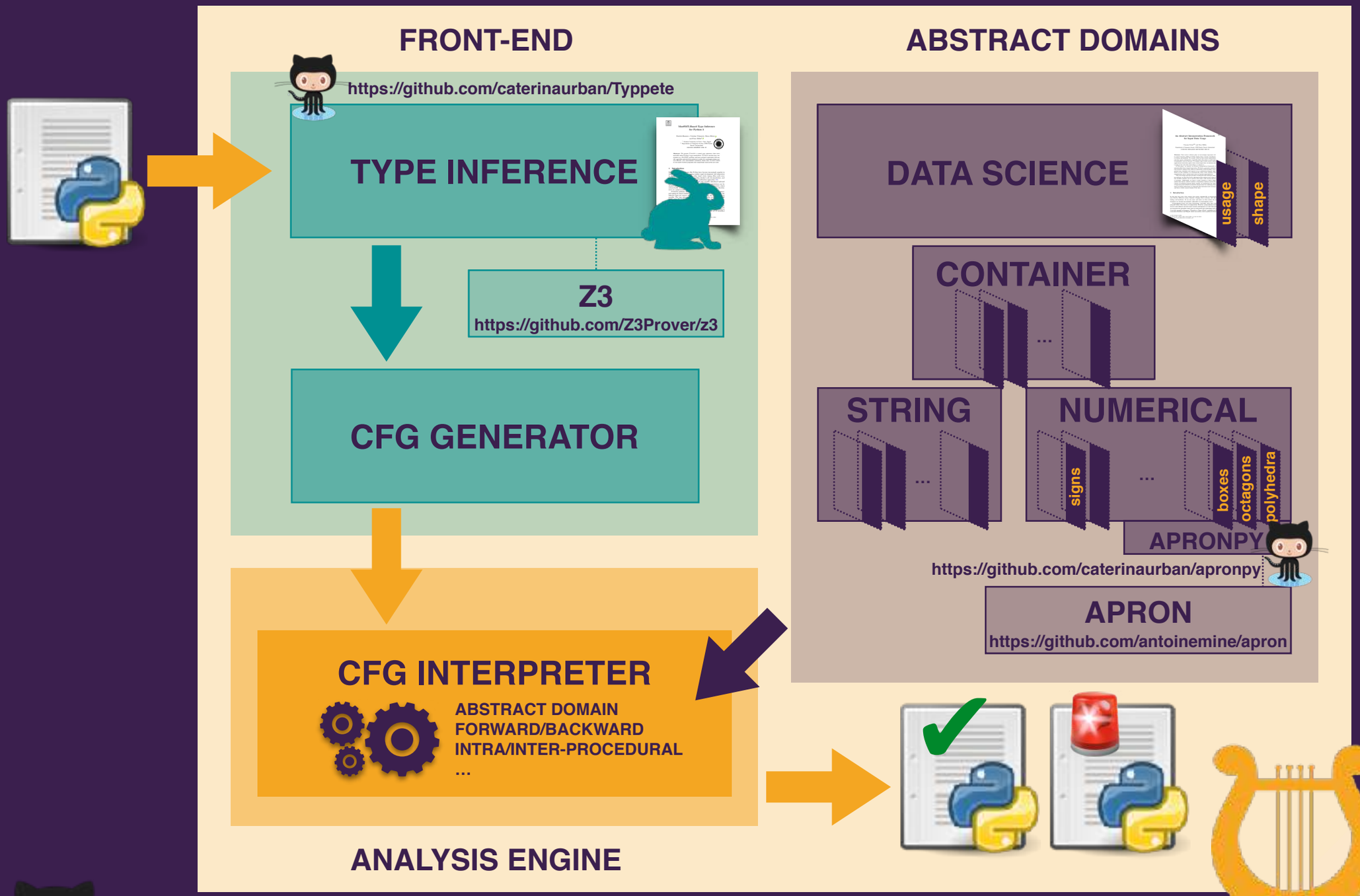
```
A C
F
dle 3 C C C
bk2 1 B+
wwb 2 D F
```

```
dle 3 C C C
bk2 1 B
wwb 1 D
```

The Lyra Static Analyzer



The Lyra Static Analyzer for Data Science Software



QUESTIONS?

https://github.com/caterinaurban/Lyra