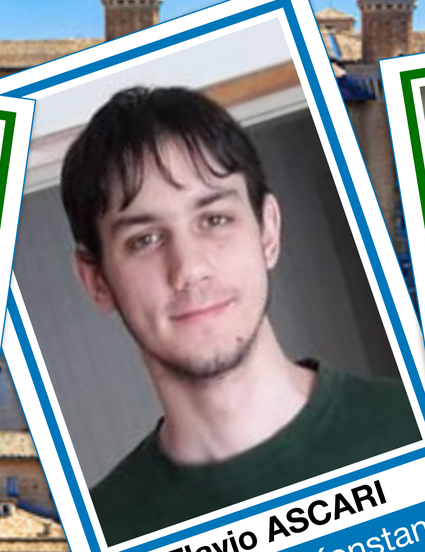
A Logic for all Reasons

THANKS FOR THE INVITATION!

Roberto BRUNI
University Pisa
Italy

Lorenzo GAZZELLA
University Pisa
Italy

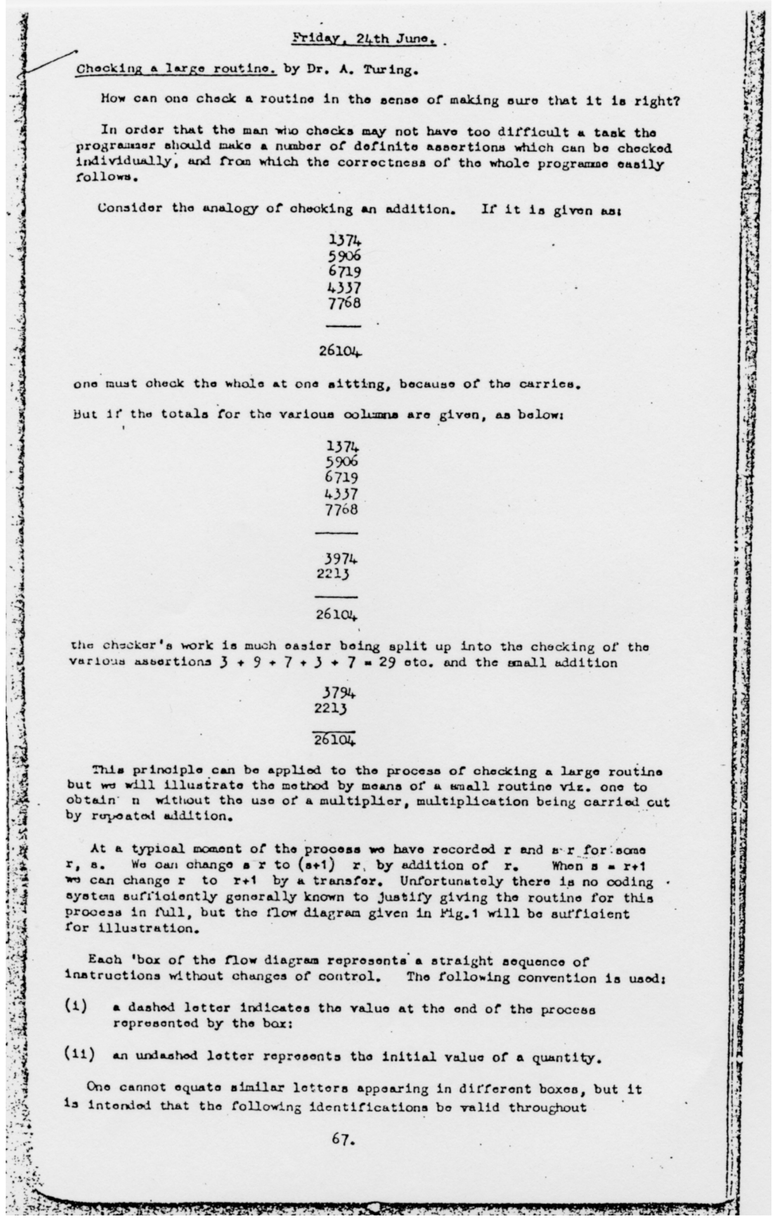


Flavio ASCARI
Universität Konstanz
Germany

Roberta GORI
University Pisa
Italy

DisCoTec 2026 - 21st International Federated Conference on Distributed Computing Techniques
June 8-12 2026, Urbino, Italy

1949: Turing's question



Friday, 24th June.

Checking a large routine. by Dr. A. Turing.

How can one check a routine in the sense of making sure that it is right?

In order that the man who checks may not have too difficult a task the programmer should make a number of definite assertions which can be checked individually, and from which the correctness of the whole programme easily follows.

Consider the analogy of checking an addition. If it is given as:

1374
5906
6719
4337
7768

26104

one must check the whole at one sitting, because of the carries.

But if the totals for the various columns are given, as below:

1374
5906
6719
4337
7768

3974
2213

26104

the checker's work is much easier being split up into the checking of the various assertions 3 + 9 + 7 + 3 + 7 = 29 etc. and the small addition

3794
2213

26104

This principle can be applied to the process of checking a large routine but we will illustrate the method by means of a small routine viz. one to obtain n without the use of a multiplier, multiplication being carried out by repeated addition.

At a typical moment of the process we have recorded r and s r for some r, s. We can change s r to (s+1) r by addition of r. When s = r+1 we can change r to r+1 by a transfer. Unfortunately there is no coding system sufficiently generally known to justify giving the routine for this process in full, but the flow diagram given in Fig.1 will be sufficient for illustration.

Each 'box of the flow diagram represents a straight sequence of instructions without changes of control. The following convention is used:

(i) a dashed letter indicates the value at the end of the process represented by the box:

(ii) an undashed letter represents the initial value of a quantity.

One cannot equate similar letters appearing in different boxes, but it is intended that the following identifications be valid throughout

67.

"How can one check a routine in the sense of making sure that it is right?"
Alan Turing (1949)

Checking factorial

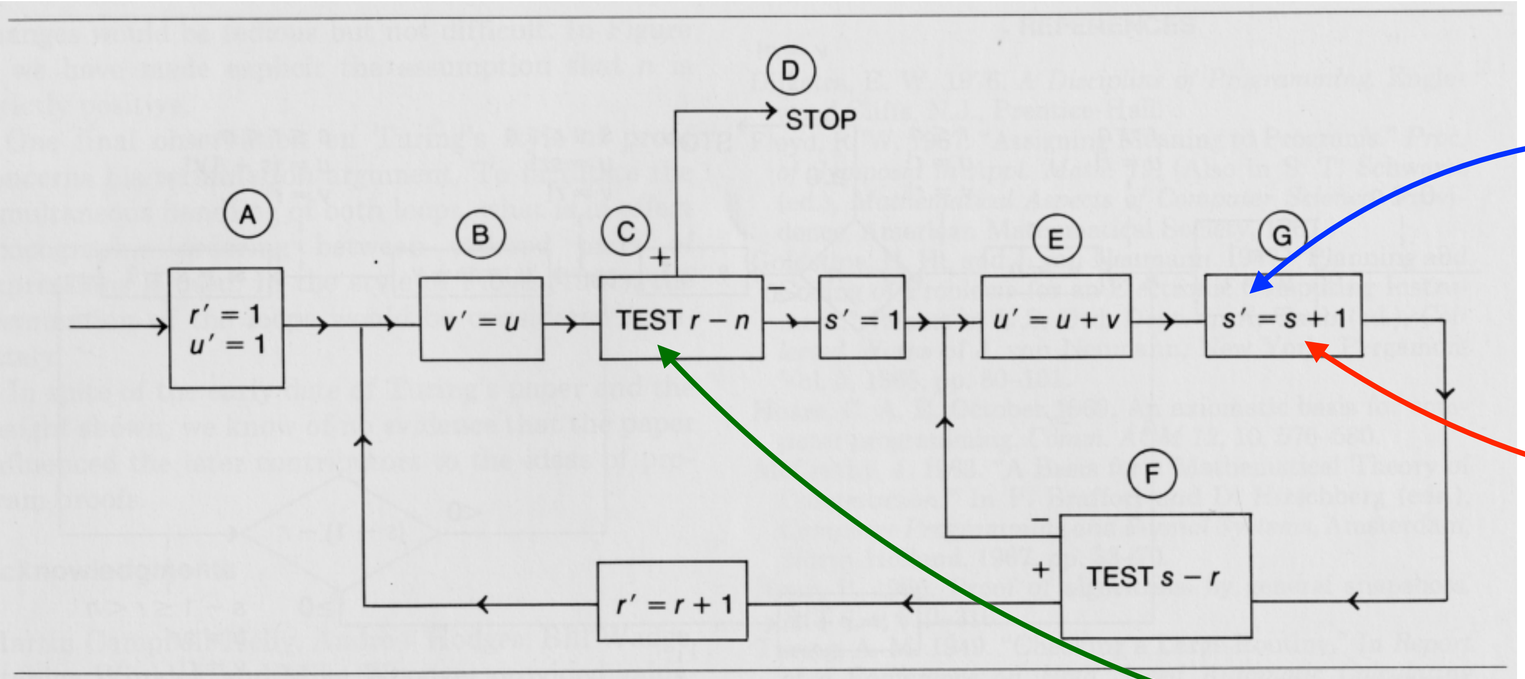


Figure 1 (Redrawn from Turing's original)

STORAGE LOCATION	(INITIAL) (A) $k = 6$	(B) $k = 5$	(C) $k = 4$	(STOP) (D) $k = 0$	(E) $k = 3$	(F) $k = 1$	(G) $k = 2$
27 **s**					s	$s + 1$	s
28 **r**		r	r		r	r	r
29 **n**	n	n	n	n	n	n	n
30 **u**		$\lfloor r$	$\lfloor r$		$s\lfloor r$	$(s+1)\lfloor r$	$(s+1)\lfloor r$
31 **v**			$\lfloor r$	$\lfloor n$	$\lfloor r$	$\lfloor r$	$\lfloor r$
	TO (B) WITH $r' = 1$ $u' = 1$	TO (C)	TO (D) IF $r = n$ TO (E) IF $r < n$		TO (G)	TO (B) WITH $r' = r + 1$ IF $s \geq r$ TO (E) WITH $s' = s + 1$ IF $s < r$	TO (F)

Figure 2 (Redrawn from Turing's original)

- a dashed letter indicates the value at the end of the process represented by the box
- an undashed letter represents the initial value of a quantity
- TEST is test for zero
- $\lfloor$_ denotes factorial
- at the end (D) $v = n!$

1969: Program correctness

Formalism for the specification and verification of program behavior, based on **pre** and **post** conditions

$$\{x \geq 0\} \quad r \quad \{x \geq 0\}$$

if the pre holds initially

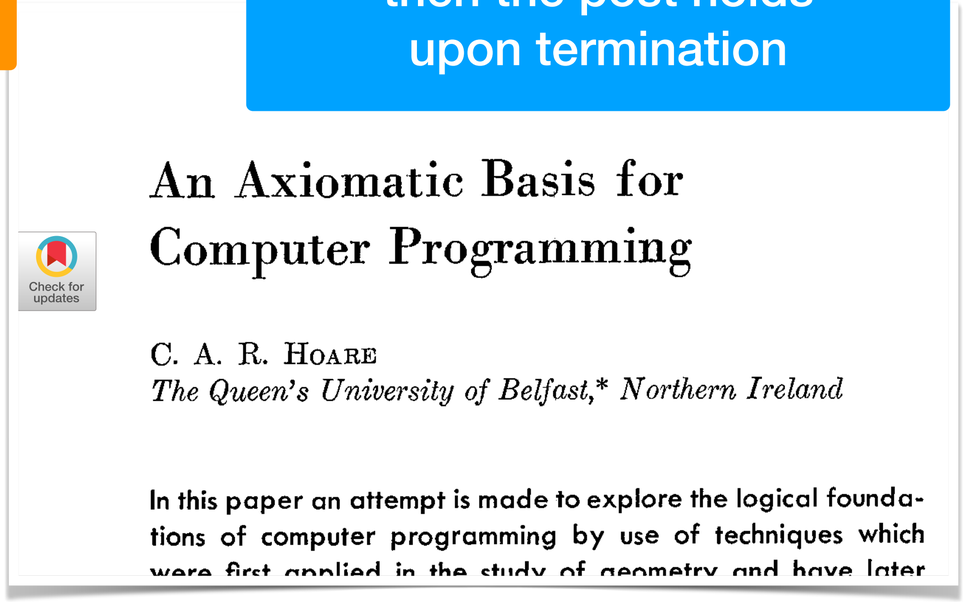
a very large and complicated program

then the post holds upon termination

Original notation:

$$P\ \{r\}\ Q$$

(1934 - 2026)



Check for updates

An Axiomatic Basis for Computer Programming

C. A. R. Hoare
The Queen's University of Belfast, Northern Ireland*

In this paper an attempt is made to explore the logical foundations of computer programming by use of techniques which were first applied in the study of geometry and have later

Hoare's example

find the quotient q and the remainder r obtained on dividing x by y

$$((r := x;\quad q := 0);\quad \textbf{while}\ y \leqslant r\ \textbf{do}\ (r := r - y;\quad q := 1 + q))$$

$$\neg y \leqslant r \wedge x = r + y \times q$$

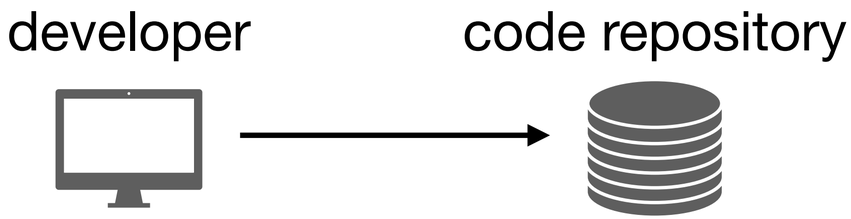
TABLE III

Line number	Formal proof	Justification
1	$\textbf{true} \supset x = x + y \times 0$	Lemma 1
2	$x = x + y \times 0\{r := x\}x = r + y \times 0$	D0
3	$x = r + y \times 0\ \{q := 0\}\ x = r + y \times q$	D0
4	$\textbf{true}\ \{r := x\}\ x = r + y \times 0$	D1 (1, 2)
5	$\textbf{true}\ \{r := x;\ \ q := 0\}\ x = r + y \times q$	D2 (4, 3)
6	$x = r + y \times q \wedge y \leqslant r \supset x = (r-y) + y \times (1+q)$	Lemma 2
7	$x = (r-y) + y \times (1+q)\{r := r-y\}x = r + y \times (1+q)$	D0
8	$x = r + y \times (1+q)\{q := 1+q\}x = r + y \times q$	D0
9	$x = (r-y) + y \times (1+q)\{r := r-y;\ q := 1+q\}\ x = r + y \times q$	D2 (7, 8)
10	$x = r + y \times q \wedge y \leqslant r\ \{r := r-y;\ q := 1+q\}\ x = r + y \times q$	D1 (6, 9)
11	$x = r + y \times q\ \{\textbf{while}\ y \leqslant r\ \textbf{do}\ (r := r-y;\ \ q := 1+q)\}$ $\neg y \leqslant r \wedge x = r + y \times q$	D3 (10)
12	$\textbf{true}\ \{((r := x;\ \ q := 0);\ \ \textbf{while}\ y \leqslant r\ \textbf{do}\ (r := r-y;\ \ q := 1+q))\}\ \neg y \leqslant r \wedge x = r + y \times q$	D2 (5, 11)

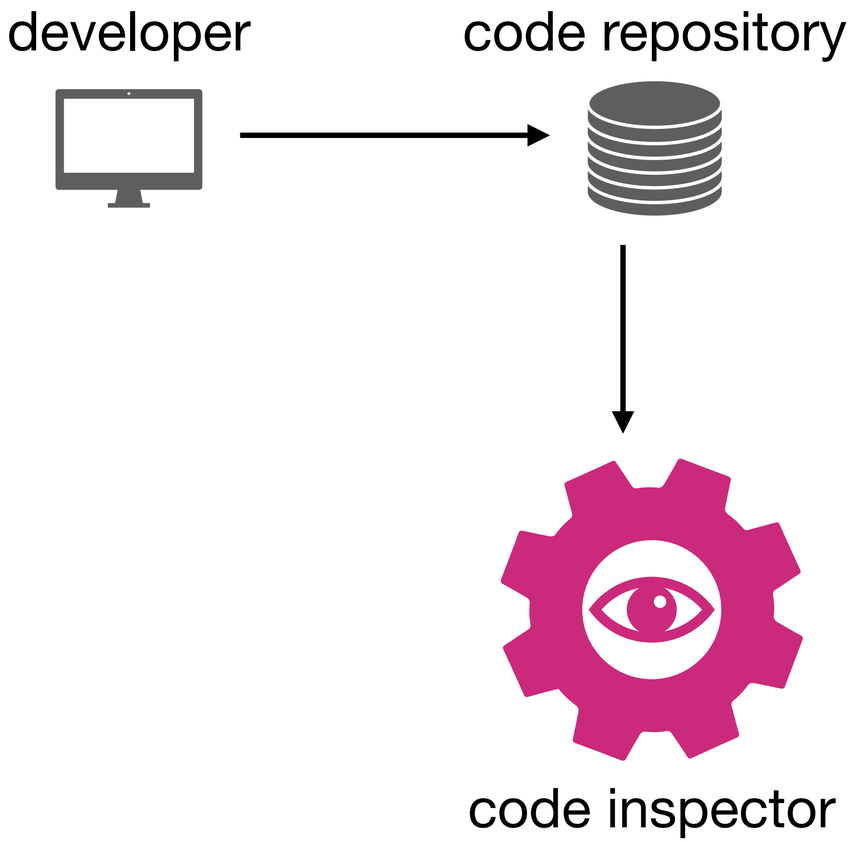
NOTES

1. The left hand column is used to number the lines, and the right hand column to justify each line, by appealing to an axiom, a lemma or a rule of inference applied to one or two previous lines, indicated in brackets. Neither of these columns is part of the formal proof. For example, line 2 is an instance of the axiom of assignment (D0); line 12 is obtained from lines 5 and 11 by application of the rule of composition (D2).
2. Lemma 1 may be proved from axioms A7 and A8.
3. Lemma 2 follows directly from the theorem proved in Sec. 2.

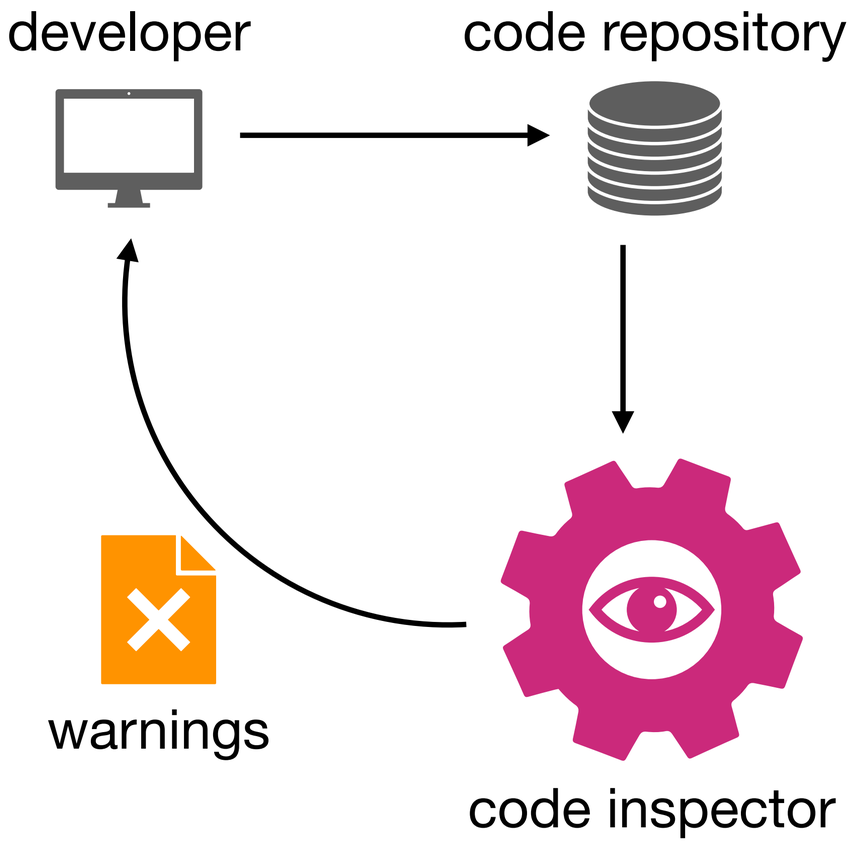
Correctness workflow, ideally



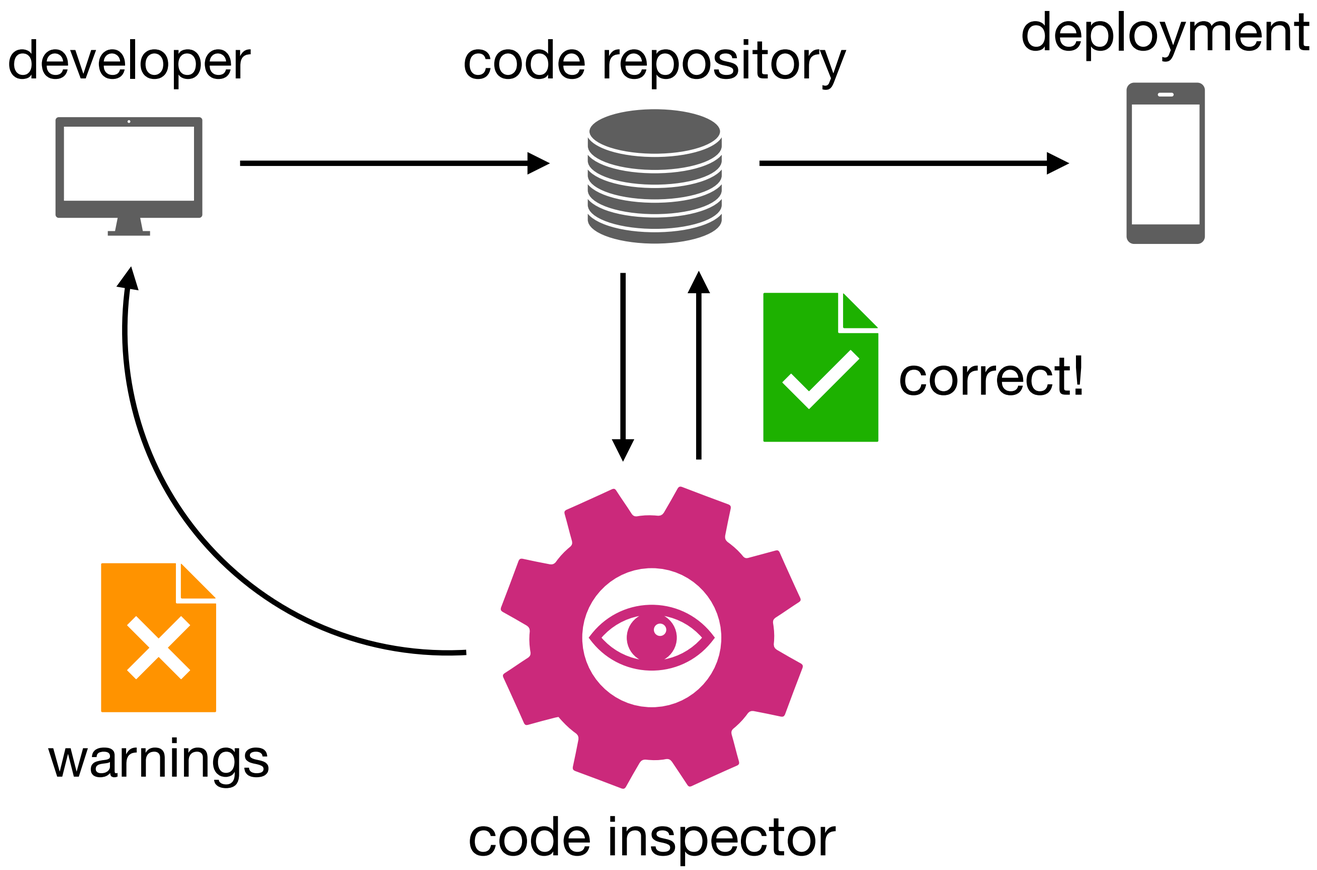
Correctness workflow, ideally



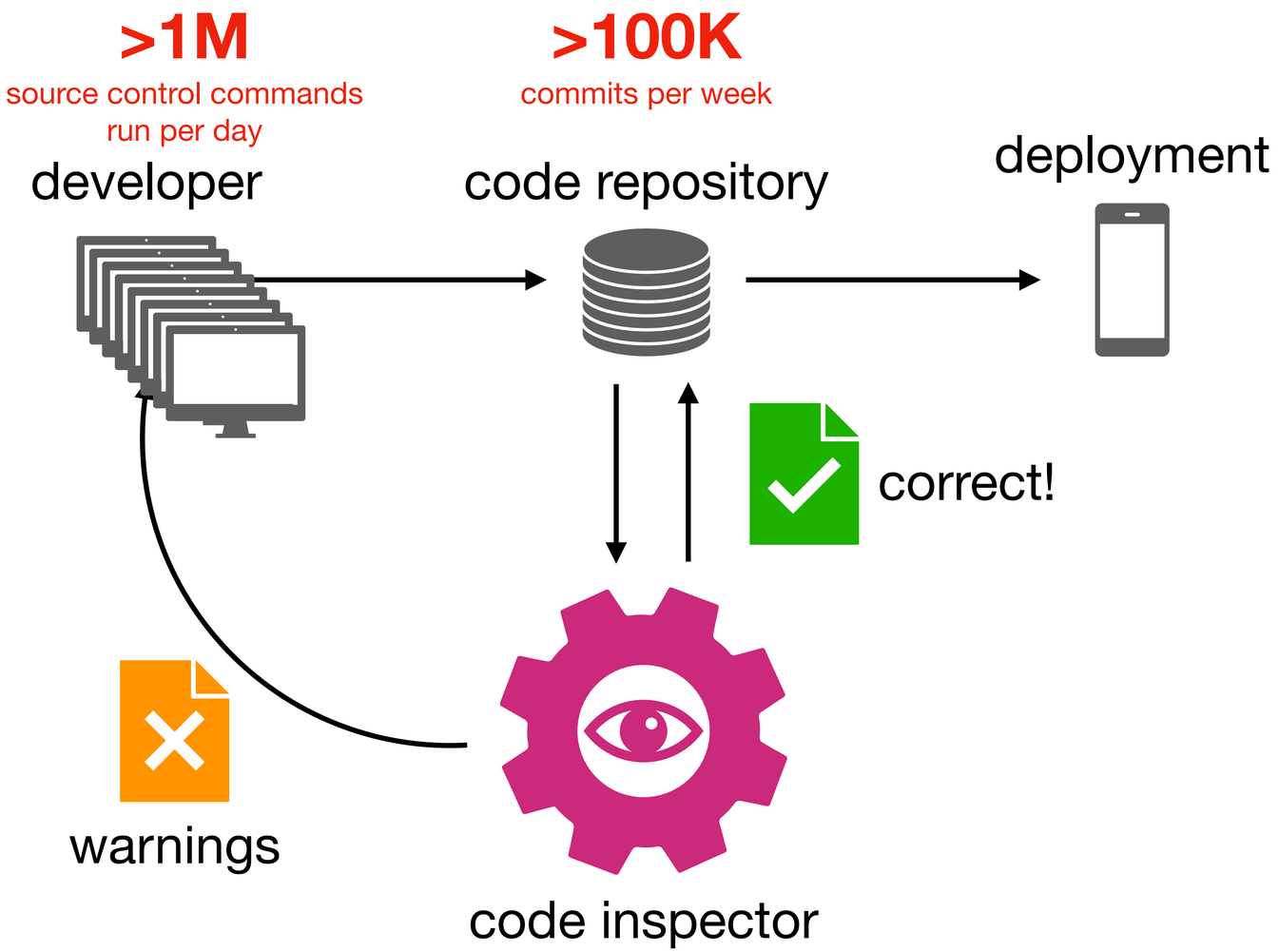
Correctness workflow, ideally



Correctness workflow, ideally

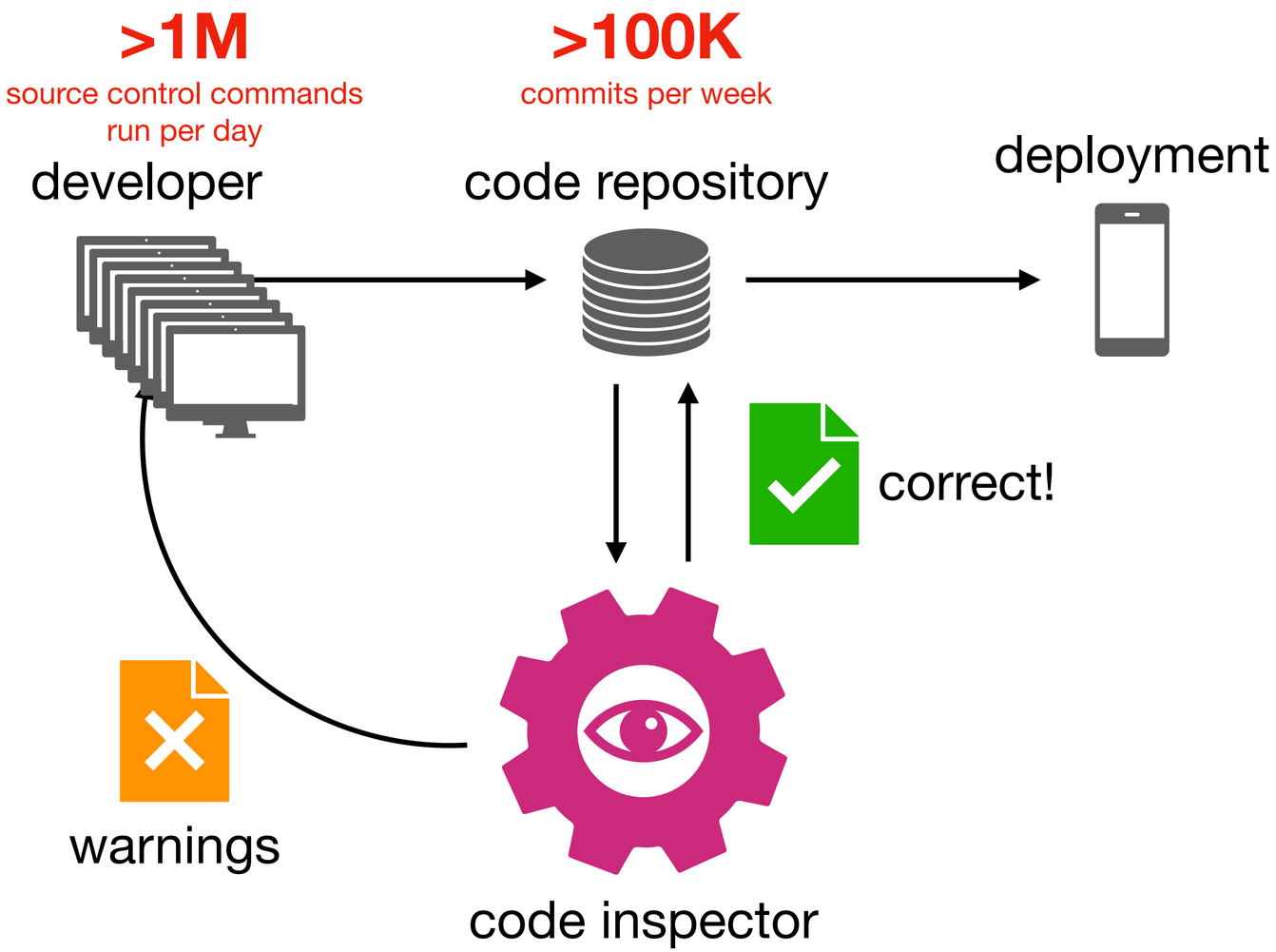


Correctness workflow, ideally



Scalability issues in a production environment:
analysis takes time (overnight?),
warnings are received late,
false positives mine credibility

Correctness workflow, ideally

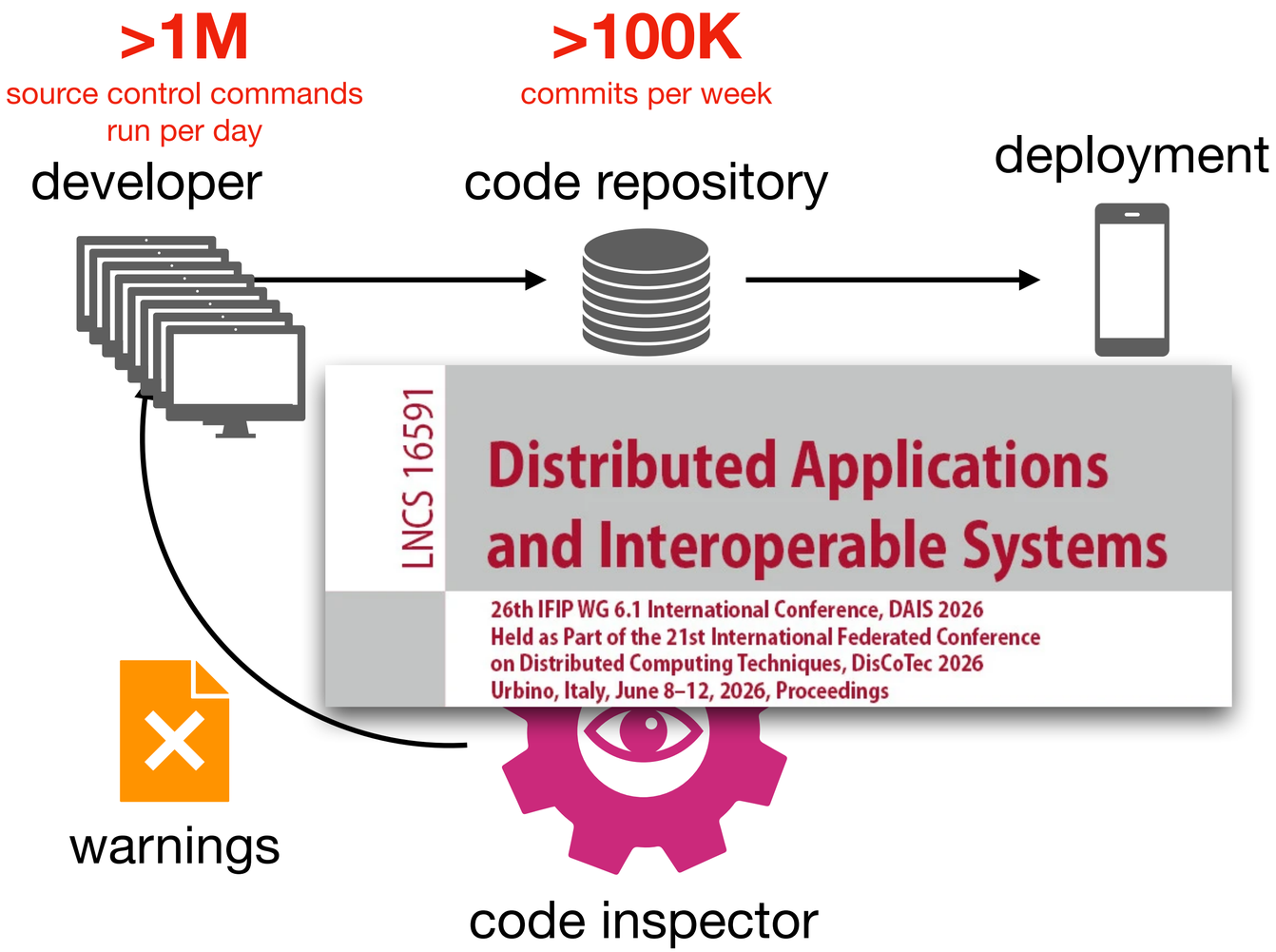


Scalability issues in a production environment:
analysis takes time (overnight?),
warnings are received late,
false positives mine credibility

"do not spam the developers!"

True positive theorem!
(under certain assumptions) the analyzer reports no false positives

Correctness workflow, ideally



Scalability issues in a production environment:
analysis takes time (overnight?),
warnings are received late,
false positives mine credibility

"do not spam the developers!"

True positive theorem!
(under certain assumptions) the analyzer reports no false positives

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation → Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$$\{P\}\ r\ \{Q\}$$
$$[P]\ r\ [Q]$$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

An Axiomatic Bas
Computer Progran

C. A.
The Q

In this
tions
were

Check for updates

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College London, UK

Program correctness and incorrectness are two sides of the same coin. As a programmer, even if you would like to have correctness, you might find yourself spending most of your time reasoning about incorrectness. This includes informal reasoning that people do while looking at or thinking about their code, as well as that supported by automated testing and static analysis tools. This paper describes a simple logic for program incorrectness which is, in a sense, the other side of the coin to Hoare's logic of correctness.

10

CCS Concepts: • **Theory of computation → Programming logic**.

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation → Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $(P)\ r\ (Q)$

$[P]\ r\ [Q]$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

An Axiomatic Bas
Computer Progran

C. A.
The Q

In this
tions
were

Check for updates

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College London, UK

Program correctness and incorrectness are two sides of the same coin. As a programmer, even if you would like to have correctness, you might find yourself spending most of your time reasoning about incorrectness. This includes informal reasoning that people do while looking at or thinking about their code, as well as that supported by automated testing and static analysis tools. This paper describes a simple logic for program incorrectness which is, in a sense, the other side of the coin to Hoare's logic of correctness.

10

CCS Concepts: • **Theory of computation → Programming logic**.

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation** → **Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $[P]\ r\ [Q]$ $(P)\ r\ (Q)$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

An Axiomatic Bas
Computer Progran

C. A.
The Q

In this
tions
were

Incorrectness Logic

PETER W. O'HEARN, Facebook and Univ

Program correctness and incorrectness are tw
like to have correctness, you might find yours
This includes informal reasoning that people d
supported by automated testing and static an
incorrectness which is, in a sense, the other si

CCS Concepts: • **Theory of computation** →

A Correctness and Incorrectness Program Logic

ROBERTO BRUNI, University of Pisa, Italy
ROBERTO GIACOBAZZI, University of Verona, Italy
ROBERTA GORI, University of Pisa, Italy
FRANCESCO RANZATO, University of Padova, Italy

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation** → **Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $[P]\ r\ [Q]$ $(P)\ r\ (Q)$ $\langle P\rangle\ r\ \langle Q\rangle$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

An Axiomatic Bas
Computer Progran

C. A.
The Q

In this
tions
were

Check for updates

Incorrectness Logic

PETER W. O'HEARN, Facebook and Univ

Program correctness and incorrectness are tw
like to have correctness, you might find yours
This includes informal reasoning that people d
supported by automated testing and static an
incorrectness which is, in a sense, the other si

CCS Concepts: • **Theory of computation** →

A Correctnes

ROBERTO BRUNI, University of Pisa, Italy
ROBERTO GIACOBAZZI, University of Verona, Italy
ROBERTA GORI, University of Pisa, Italy
FRANCESCO RANZATO, University of Padova, Italy

Outcome Logic: A Unifying Foundation for Correctness and Incorrectness Reasoning

NOAM ZILBERSTEIN, Cornell University, USA
DEREK DREYER, MPI-SWS, Germany
ALEXANDRA SILVA, Cornell University, USA

Program logics for bug-finding (such as the recently introduced Incorrectness Logic) have framed correctness and incorrectness as dual concepts requiring different logical foundations. In this paper, we argue that a single

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation** → **Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $[P]\ r\ [Q]$ $(P)\ r\ (Q)$ $\langle P\rangle\ r\ \langle Q\rangle$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

An Axiomatic Bas
Computer Progran

C. A.
The Q

In this
tions
were

Incorrectness Logic

PETER W. O'HEARN, Facebook and Univ

Program correctness and incorrectness are tw
like to have correctness, you might find yours
This includes informal reasoning that people d
supported by automated testing and static an
incorrectness which is, in a sense, the other si

CCS Concepts: • **Theory of computation** →

A Correctnes

ROBERTO BRUNI, University of Pisa, Italy
ROBERTO GIACOBAZZI, University of Verona, Italy
ROBERTA GORI, University of Pisa, Italy
FRANCESCO RANZATO, University of Padova, Italy

Outcome Logic: A Unifying
Incorrectness Reasoning

NOAM ZILBERSTEIN, Cornell University, US
DEREK DREYER, MPI-SWS, Germany
ALEXANDRA SILVA, Cornell University, USA

Program logics for bug-finding (such as the recentl
and incorrectness as dual concepts requiring differe

Check for updates

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

Exact Separation Logic

Towards Bridging the Gap Between Verification and Bug-Finding

Petar Maksimović
Imperial College London, UK
Runtime Verification Inc., Urbana, IL, USA

Caroline Cronjäger
Ruhr-Universität Bochum, Germany

Andreas Lööw
Imperial College London, UK

Julian Sutherland
Nethermind, London, UK

Philippa Gardner
Imperial College London, UK

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation → Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $[P]\ r\ [Q]$ $(P)\ r\ (Q)$ $\langle P\rangle\ r\ \langle Q\rangle$ $\langle\langle P\rangle\rangle\ r\ \langle\langle Q\rangle\rangle$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

An Axiomatic Bas
Computer Progran

C. A.
The Q

In this
tions
were

Check for updates

Incorrectness Logic

PETER W. O'HEARN, Facebook and Univ

Program correctness and incorrectness are tw
like to have correctness, you might find yours
This includes informal reasoning that people d
supported by automated testing and static an
incorrectness which is, in a sense, the other si

CCS Concepts: • **Theory of computation →**

A Correctnes

ROBERTO BRUNI, University of Pisa, Italy
ROBERTO GIACOBAZZI, University of Verona, Italy
ROBER
FRANC

Outcome Logic: A Unifying
Incorrectness Reasoning

NOAM ZILBERSTEIN, Cornell University, US
DEREK DREYER, MPI-SWS, Germany
ALEXANDRA SILVA, Cornell University, USA

Program logics for bug-finding (such as the recentl
and incorrectness as dual concepts requiring differe

Exact Separation Logic

Towards Bridging the Gap Between Verification and Bug-Finding

Petar Maksimović
Imperial College London, UK
Runtime Verification Inc., Urbana, IL, USA

Caroline Cronjäger
Ruhr-Universität Bochum, Germany

Andreas Lööw
Imperial College London, UK

d
K

r
, UK

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

Revealing Sources of (Memory) Errors via Backward Analysis

FLAVIO ASCARI, University of Pisa, Italy
ROBERTO BRUNI, University of Pisa, Italy
ROBERTA GORI, University of Pisa, Italy
FRANCESCO LOGOZZO, Meta Platforms, USA

Sound over-approximation methods are effective for proving the absence of errors, but inevitably produce false alarms that can hamper programmers. In contrast, under-approximation methods focus on bug detection

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation → Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $[P]\ r\ [Q]$ $(P)\ r\ (Q)$ $\langle P\rangle\ r\ \langle Q\rangle$ $\langle\langle P\rangle\rangle\ r\ \langle\langle Q\rangle\rangle$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

An Axiomatic Bas
Computer Progran

C. A.
The Q

In this
tions
were

Incorrectness Logic

PETER W. O'HEARN, Facebook and Univ

Program correctness and incorrectness are tw
like to have correctness, you might find yours
This includes informal reasoning that people d
supported by automated testing and static an
incorrectness which is, in a sense, the other si

CCS Concepts: • **Theory of computation →**

A Correctnes

ROBERTO BRUNI, University of Pisa, Italy
ROBERTO GIACOBAZZI, University of Verona, Italy
ROBER
FRANC

Outcome Logic: A Unifying
Incorrectness Reasoning

NOAM ZILBERSTEIN, Cornell University, US
DEREK DREYER, MPI-SWS, Germany
ALEXANDRA SILVA, Cornell University, USA

Program logics for bug-finding (such as the recentl
and incorrectness as dual concepts requiring differe

Exact Separation Logic

Towards Bridging the Gap Between Verification and Bug-Finding

Petar Maksimović
Imperial College London, UK
Runtime Verification Inc., Urbana, IL, USA

Caroline Cronjäger
Ruhr-Universität Bochum, Germany

Andreas Lööw
Imperial College London, UK

Check for updates

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

Revealing Sources of (Memory) Errors via Backward Analysis

FLAVIO ASCARI, University of Pisa, Italy
ROBERTO BRUNI, University of Pisa, Italy
ROBERTA GORI, University of Pisa, Italy
FRANCESCO LOGOZZO, Meta Platforms, USA

Sound over-approximation methods are effective fo
false alarms that can hamper programmers. In contra

On Extending Incorrectness Logic with Backwards Reasoning

FREEK VERBEEK, Open Universiteit of the Netherlands, Netherlands and Virginia Tech, USA
MD SYADUS SEFAT, Virginia Tech, USA
ZHOULAI FU, State University of New York, Republic of Korea, Stony Brook University, USA, and Virginia Tech, USA
BINOY RAVINDRAN, Virginia Tech, USA

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation** → **Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $[P]\ r\ [Q]$ $(P)\ r\ (Q)$ $\langle P\rangle\ r\ \langle Q\rangle$ $\langle\langle P\rangle\rangle\ r\ \langle\langle Q\rangle\rangle$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

A Taxonomy of Hoare-Like Logics

Towards a Holistic View using Predicate Transformers and Kleene Algebras with Top and Tests

LENA VERSCHT, Saarland University, Germany and RWTH Aachen, Germany
BENJAMIN LUCIEN KAMINSKI, Saarland University, Germany and University College London, UK

Outcome Logic: A Unifyin
Incorrectness Reasoning

NOAM ZILBERSTEIN, Cornell University, US
DEREK DREYER, MPI-SWS, Germany
ALEXANDRA SILVA, Cornell University, USA

Program logics for bug-finding (such as the recentl
and incorrectness as dual concepts requiring differe

Petar Maksimović
Imperial College London, UK
Runtime Verification Inc., Urbana, IL, USA

Caroline Cronjäger
Ruhr-Universität Bochum, Germany

Andreas Lööw
Imperial College London, UK

An Axiomatic Bas
Computer Progran

C. A.
The Q

In this
tions
were

Check for updates

Incorrectness Logic

PETER W. O'HEARN, Facebook and Univ

Program correctness and incorrectness are tw
like to have correctness, you might find yours
This includes informal reasoning that people d
supported by automated testing and static an
incorrectness which is, in a sense, the other si

CCS Concepts: • **Theory of computation** →

A Correctnes

ROBERTO BRUNI, University of Pisa, Italy
ROBERTO GIACOBAZZI, University of Verona, Italy
ROBER
FRANC

Revealing Sources of (Memory) Errors via Backward Analysis

FLAVIO ASCARI, University of Pisa, Italy
ROBERTO BRUNI, University of Pisa, Italy
ROBERTA GORI, University of Pisa, Italy
FRANCESCO LOGOZZO, Meta Platforms, USA

Sound over-approximation methods are effective fo
false alarms that can hamper programmers. In contra

On Extending Incorrectness Logic with Backwards Reasoning

FREEK VERBEEK, Open Universiteit of the Netherlands, Netherlands and Virginia Tech, USA
MD SYADUS SEFAT, Virginia Tech, USA
ZHOULAI FU, State University of New York, Republic of Korea, Stony Brook University, USA, and Virginia Tech, USA
BINOY RAVINDRAN, Virginia Tech, USA

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation** → **Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $[P]\ r\ [Q]$ $(P)\ r\ (Q)$ $\langle P\rangle\ r\ \langle Q\rangle$ $\langle\langle P\rangle\rangle\ r\ \langle\langle Q\rangle\rangle$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

A Taxonomy of Hoare-Like Logics

Towards a Holistic View using Predicate Transformers and Kleene Algebras with Top and Tests

LENA VERSCHT, Saarland University, Germany and RWTH Aachen, Germany
BENJAMIN LUCIEN KAMINSKI, Saarland University, Germany and University College London, UK

Calculational Design of [In]Correctness Transformational Program Logics by Abstract Interpretation

PATRICK COUSOT, New York University, USA

An Axiomatic Bas
Computer Progran

C. A.
The Q

In this
tions
were

Check for updates

Incorrectne

PETER W. O'HEA

Program correctness and incorrectness are tw
like to have correctness, you might find yours
This includes informal reasoning that people d
supported by automated testing and static an
incorrectness which is, in a sense, the other si

CCS Concepts: • **Theory of computation** →

ROBERTO BRUNI, University of Pisa, Italy
ROBERTO GIACOBAZZI, University of Verona, Italy
ROBER
FRANC

Outcome Logic: A Unifyin

simović
London, UK
ation Inc., Urbana, IL, USA

ronjäger
Ruhr-Universitat Bochum, Germany

Andreas Lööw
Imperial College London, UK

d

Revealing Sources of (Memory) Errors via Backward Analysis

FLAVIO ASCARI, University of Pisa, Italy
ROBERTO BRUNI, University of Pisa, Italy
ROBERTA GORI, University of Pisa, Italy
FRANCESCO LOGOZZO, Meta Platforms, USA

Sound over-approximation methods are effective fo
false alarms that can hamper programmers. In contra

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

On Extending Incorrectness Logic with Backwards Reasoning

FREEK VERBEEK, Open Universiteit of the Netherlands, Netherlands and Virginia Tech, USA
MD SYADUS SEFAT, Virginia Tech, USA
ZHOULAI FU, State University of New York, Republic of Korea, Stony Brook University, USA, and Virginia Tech, USA
BINOY RAVINDRAN, Virginia Tech, USA

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation** → **Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $[P]\ r\ [Q]$ $(P)\ r\ (Q)$ $\langle P\rangle\ r\ \langle Q\rangle$ $\langle\langle P\rangle\rangle\ r\ \langle\langle Q\rangle\rangle$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

A Taxonomy of Hoare-Like Logics

Towards a Holistic View using Predicate Transformers and Kleene Algebras with Top and Tests

LENA VERSCHT, Saarland University, Germany and RWTH Aachen, Germany
BENJAMIN LUCIEN KAMINSKI, Saarland University, Germany and University College London, UK

Calculational Design of [In]Correctness Transformational Program Logics by Abstract Interpretation

PATRICK COUSOT, New York University, USA

ROBERTO BRUNI, University of Pisa, Italy
ROBERTO GIACOBAZZI, University of Verona, Italy

U-Turn: Enhancing Incorrectness Analysis by Reversing Direction

FLAVIO ASCARI, University of Konstanz, Germany
ROBERTO BRUNI, University of Pisa, Italy
ROBERTA GORI, University of Pisa, Italy
AZALEA RAAD, Imperial College London, United Kingdom

Revealing Sources of (Memory) Errors via Ba

FLAVIO ASCARI, University of Pisa, Italy
ROBERTO BRUNI, University of Pisa, Italy
ROBERTA GORI, University of Pisa, Italy
FRANCESCO LOGOZZO, Meta Platforms, USA

Sound over-approximation methods are effective fo
false alarms that can hamper programmers. In contra

On Exten

FREEK VERB
MD SYADUS
ZHOULAI FU, State University of New York, Republic of Korea, Stony Brook University, USA, and Virginia Tech, USA
BINOY RAVINDRAN, Virginia Tech, USA

An Axiomatic Bas
Computer Progran

C. A.
The Q

In this
tions
were

Check for updates

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation** → **Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $[P]\ r\ [Q]$ $(P)\ r\ (Q)$ $\langle P\rangle\ r\ \langle Q\rangle$ $\langle\langle P\rangle\rangle\ r\ \langle\langle Q\rangle\rangle$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

A Taxonomy of Hoare-Like Logics

Towards a Holistic View using Predicate Transformers and Kleene Algebras with Top and Tests

LENA VERSCHT, Saarland University, Germany and RWTH Aachen, Germany
BENJAMIN LUCIEN KAMINSKI, Saarland University, Germany and University College London, UK

Calculational Design of [In]Correctness Transformational Program Logics by Abstract Interpretation

PATRICK COUSOT, New York University, USA

Non-termination Proving at Scale

AZALEA RAAD, Imperial College London, UK and Bloomberg, USA
JULIEN VANEGUE, Bloomberg, USA and Imperial College London, UK
PETER O'HEARN, University College London, UK

U-Turn: Enhancing Incorrectness Analysis by Reversing Direction

FLAVIO ASCARI, University of Konstanz, Germany
ROBERTO BRUNI, University of Pisa, Italy
ROBERTA GORI, University of Pisa, Italy
AZALEA RAAD, Imperial College London, United Kingdom

On Exten

FREEK VERB
MD SYADUS
ZHOULAI FU, State University of New York, Republic of Korea, Stony Brook University, USA, and Virginia Tech, USA
BINOY RAVINDRAN, Virginia Tech, USA

Axiomatic Approach to Total Correctness of Programs

Zohar Manna and Amir Pnueli

Received June 20, 1973

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

2020: Program incorrectness

Incorrectness Logic

PETER W. O'HEARN, Facebook and University College Londo

Program correctness and incorrectness are two sides of the same c
like to have correctness, you might find yourself spending most of
This includes informal reasoning that people do while looking at or
supported by automated testing and static analysis tools. This pa
incorrectness which is, in a sense, the other side of the coin to Hoa

CCS Concepts: • **Theory of computation → Programming logi**

Additional Key Words and Phrases: Proofs, Bugs, Static Analysis

ACM Reference Format:

$\{P\}\ r\ \{Q\}$ $[P]\ r\ [Q]$ $(P)\ r\ (Q)$ $\langle P\rangle\ r\ \langle Q\rangle$ $\langle\langle P\rangle\rangle\ r\ \langle\langle Q\rangle\rangle$

"Program correctness and incorrectness are two sides of the same coin"

Renewed interest in program logics

A Taxonomy of Hoare-Like Logics

Towards a Holistic View using Predicate Transformers and Kleene Algebras with Top and Tests

LENA VERSCHT, Saarland University, Germany and RWTH Aachen, Germany
BENJAMIN LUCIEN KAMINSKI, Saarland University, Germany and University College London, UK

Outcome Logic: A Unifyin

simović
London, UK
ation Inc., Urbana, IL, USA
ronjäger

Calculational Design of [In]Correctness Transformational Program Logics by Abstract Interpretation

Incorrectne

An Axiomatic Ba
Computer Pr

C. A.
The Q

Check for updates

Non-termination

AZALEA RAAD, Imperial C
JULIEN VANEGUE, Bloom
PETER O'HEARN, Univers

: Enhancing Incorrectness Analysis by Reversing
on

SCARI, University of Konstanz, Germany
BRUNI, University of Pisa, Italy
GORI, University of Pisa, Italy
RAAD, Imperial College London, United Kingdom

of New York, Republic of Korea, Stony Brook University, USA, and Virginia

BINOY RAVINDRAN, Virginia Tech, USA

Axiomatic Approach to Total C

Zohar Manna and An

Received June 20,

Summary. We present here an axiomatic approach which enables one to prove by formal methods that his program is "totally correct" (i.e., it terminates and is

In this
tions
were

LNCS 16589

Formal Techniques for Distributed Objects, Components, and Systems

46th IFIP WG 6.1 International Conference, FORTE 2026
Held as Part of the 21st International Federated Conference on Distributed Computing Techniques, DisCoTec 2026
Urbino, Italy, June 8–12, 2026, Proceedings

Program logics zoo

Program logics zoo

post

pre

post

wp

pre

wpp

wlp

Why3
Where Programs Meet Provers

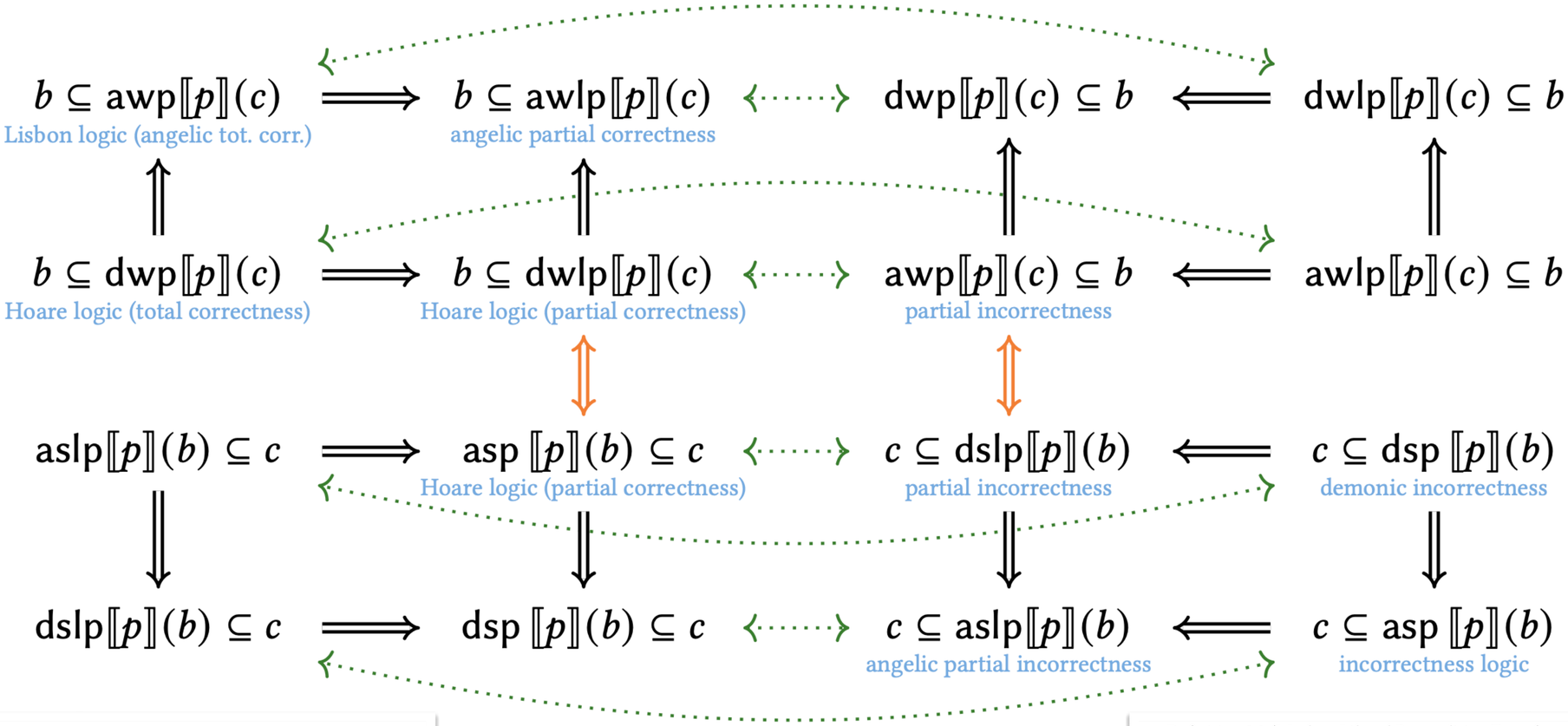
Infer

Verified Software Toolchain

Dafny

SOTERIA

Predicate transformes



Definition 3.1 (Weakest Precondition Transformers [Dijkstra 1976]*).* Given a program p and a postcondition c, the *angelic weakest precondition* is defined as

$$\mathrm{awp}[\![p]\!](c) \;=\; \lambda\sigma.\ \bigvee_{\tau \in [\![p]\!](\sigma)} c(\tau)\ .$$

The *demonic weakest precondition* is defined as

$$\mathrm{dwp}[\![p]\!](c) \;=\; \lambda\sigma.\ \begin{cases} \text{false}, & \text{if } p \text{ can diverge on } \sigma \\ \bigwedge_{\tau \in [\![p]\!](\sigma)} c(\tau), & \text{otherwise .} \end{cases}$$

◁

[POPL25] Verscht and Kaminski

Angelic vs Demonic

Liberal vs Terminating

Strongest post vs Weakest pre

Definition 3.2 (Weakest Liberal Precondition Transformers [Dijkstra 1976]*).* Given a program p and a postcondition c, the *demonic weakest liberal precondition* is defined as

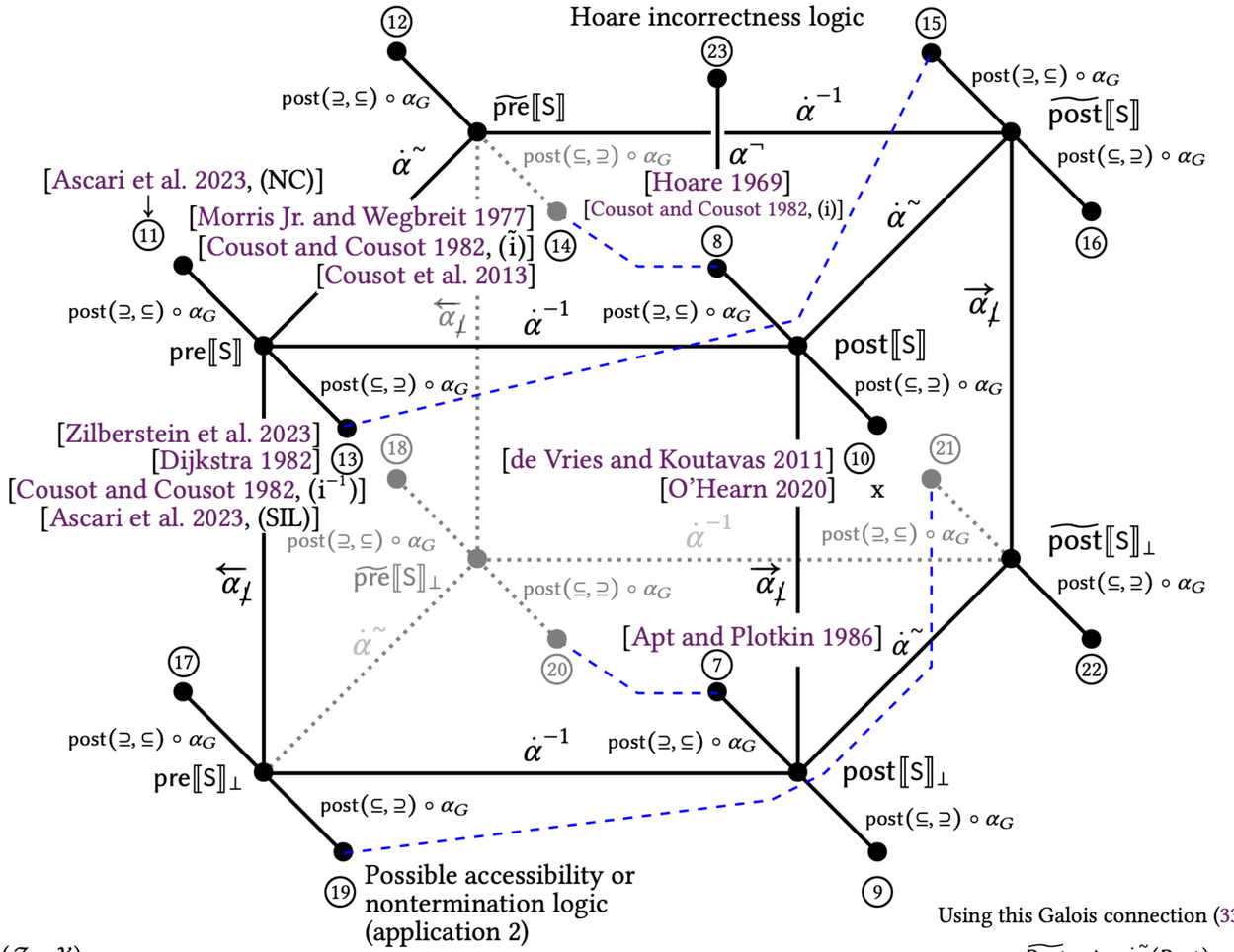
$$\mathrm{dwlp}[\![p]\!](c) \;=\; \lambda\sigma.\ \bigwedge_{\tau \in [\![p]\!](\sigma)} c(\tau)\ .$$

The *angelic weakest liberal precondition* is defined as

$$\mathrm{awlp}[\![p]\!](c) \;=\; \lambda\sigma.\ \begin{cases} \text{true}, & \text{if } p \text{ can diverge on } \sigma \\ \bigvee_{\tau \in [\![p]\!](\sigma)} c(\tau), & \text{otherwise .} \end{cases}$$

◁

Calculational design



$$\begin{array}{rcl} \mathsf{Post} & \in & \wp(\mathcal{X} \times \mathcal{Y}) \to \wp(\mathcal{Z} \times \mathcal{X}) \to \wp(\mathcal{Z} \times \mathcal{X}) \\ \mathsf{Post}(r)P & \triangleq & \{\langle \sigma_0,\, \sigma' \rangle \mid \exists \sigma \,.\, \langle \sigma_0,\, \sigma \rangle \in P \wedge \langle \sigma,\, \sigma' \rangle \in r\} \end{array} \tag{6}$$

Using these Galois isomorphisms (28), we define the precondition transformer Ⓐ

$$\mathsf{Pre} \triangleq \vec{\alpha}^{-1}(\mathsf{Post}) = \boldsymbol{\lambda} r \bullet \boldsymbol{\lambda} Q \bullet \{\langle \sigma,\, \sigma_f \rangle \mid \exists \sigma' \,.\, \langle \sigma,\, \sigma' \rangle \in r \wedge \langle \sigma',\, \sigma_f \rangle \in Q)\} \tag{29}$$

[POPL24] Cousot
Abstract Interpretation

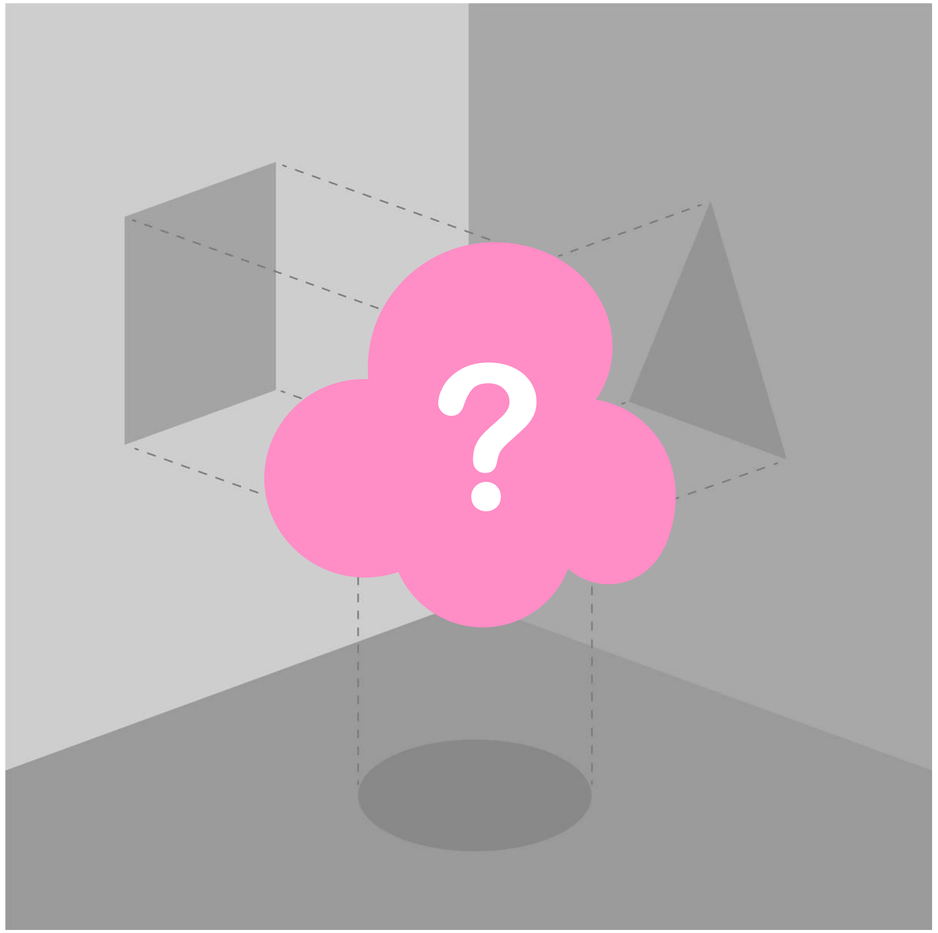
Using this Galois connection (33), we define the dual complement transformers Ⓐ

$$\begin{array}{rcl} \widetilde{\mathsf{Post}} \triangleq \dot{\alpha}^{\sim}(\mathsf{Post}) & = & \boldsymbol{\lambda} r \bullet \boldsymbol{\lambda} P \bullet \{\langle \sigma_0,\, \sigma' \rangle \mid \forall \sigma \,.\, \langle \sigma,\, \sigma' \rangle \in r \Rightarrow \langle \sigma_0,\, \sigma \rangle \in P\} \\ \widetilde{\mathsf{Pre}} \triangleq \dot{\alpha}^{\sim}(\mathsf{Pre}) & = & \boldsymbol{\lambda} r \bullet \boldsymbol{\lambda} Q \bullet \{\langle \sigma,\, \sigma_f \rangle \mid \forall \sigma' \,.\, \langle \sigma,\, \sigma' \rangle \in r \Rightarrow \langle \sigma',\, \sigma_f \rangle \in Q\} \end{array} \tag{34}$$

If $r \in \wp(\mathcal{X} \times \mathcal{Y})$ then Ⓐ

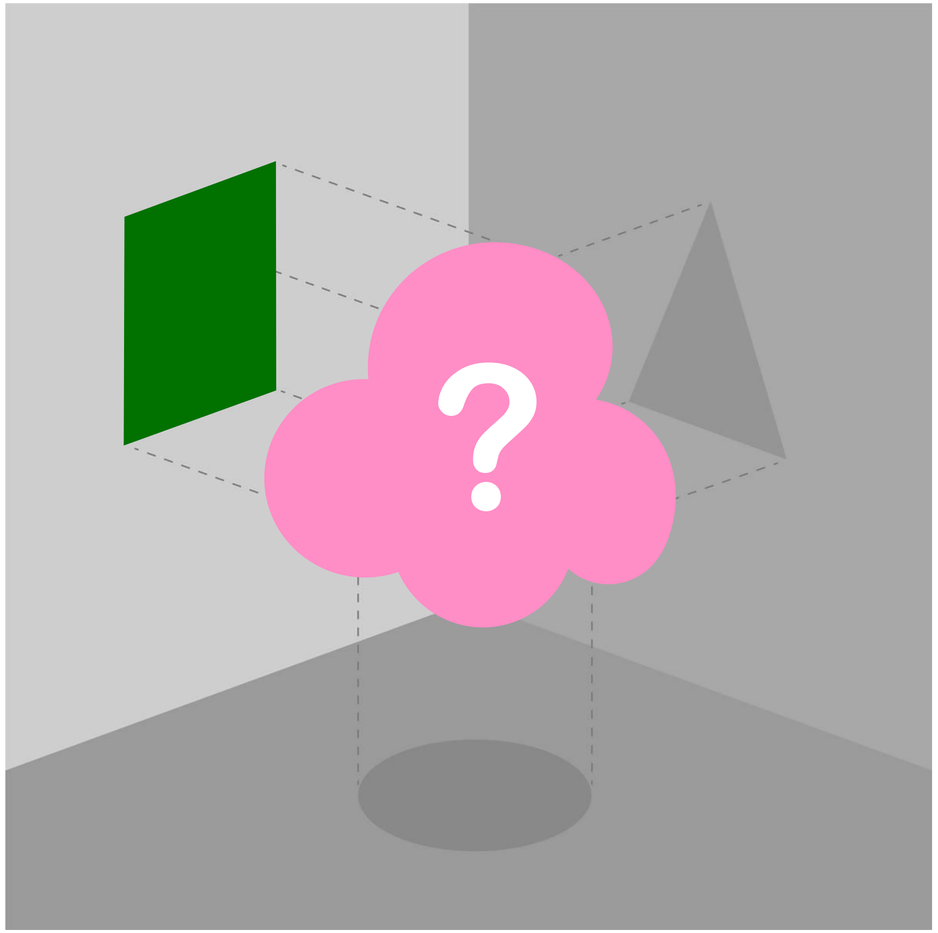
$$\langle \wp(\mathcal{Z} \times \mathcal{X}),\, \subseteq \rangle \underset{\mathsf{Post}(r)}{\overset{\dot{\alpha}^{-1}(\widetilde{\mathsf{Pre}}(r))}{\leftrightarrows}} \langle \wp(\mathcal{Z} \times \mathcal{Y}),\, \subseteq \rangle \quad \langle \wp(\mathcal{X} \times \mathcal{Z}),\, \subseteq \rangle \underset{\mathsf{Pre}(r)}{\overset{\dot{\alpha}^{-1}(\widetilde{\mathsf{Post}}(r))}{\leftrightarrows}} \langle \wp(\mathcal{Y} \times \mathcal{Z}),\, \subseteq \rangle \tag{35}$$

Problem



Each program logic illuminates different aspects but yields an incomplete view

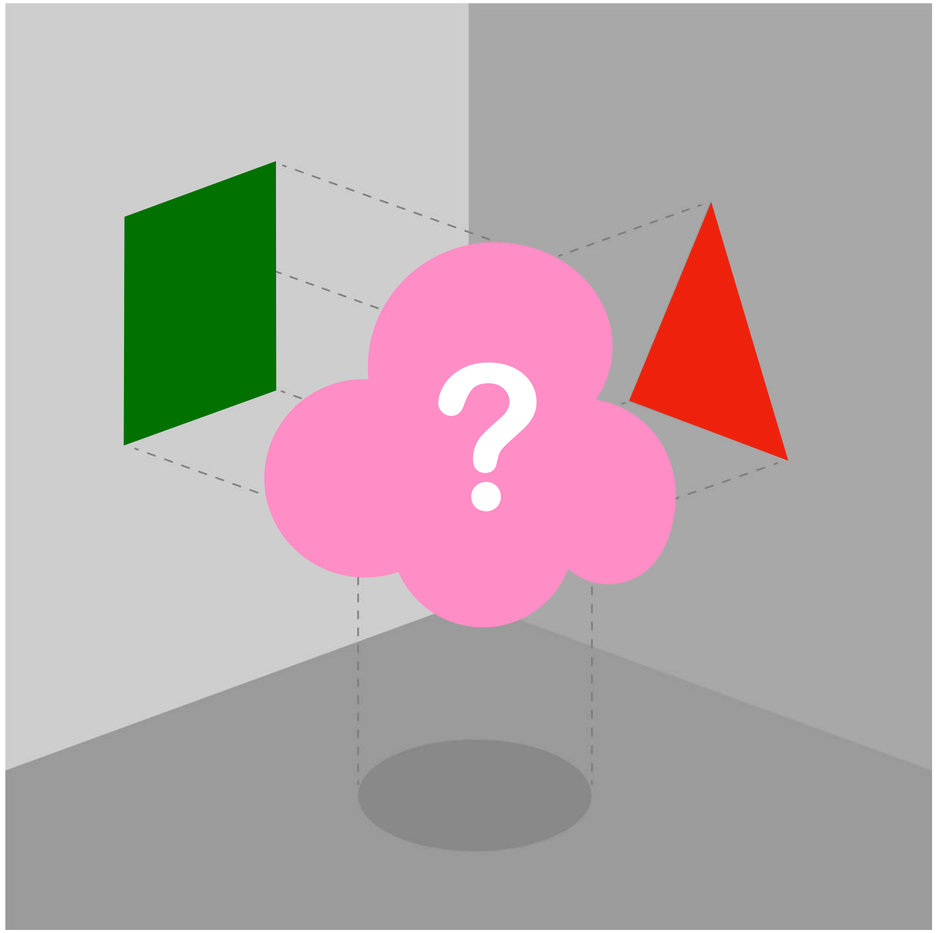
Problem



Each program logic illuminates different aspects but yields an incomplete view

- HL: Correct states on termination

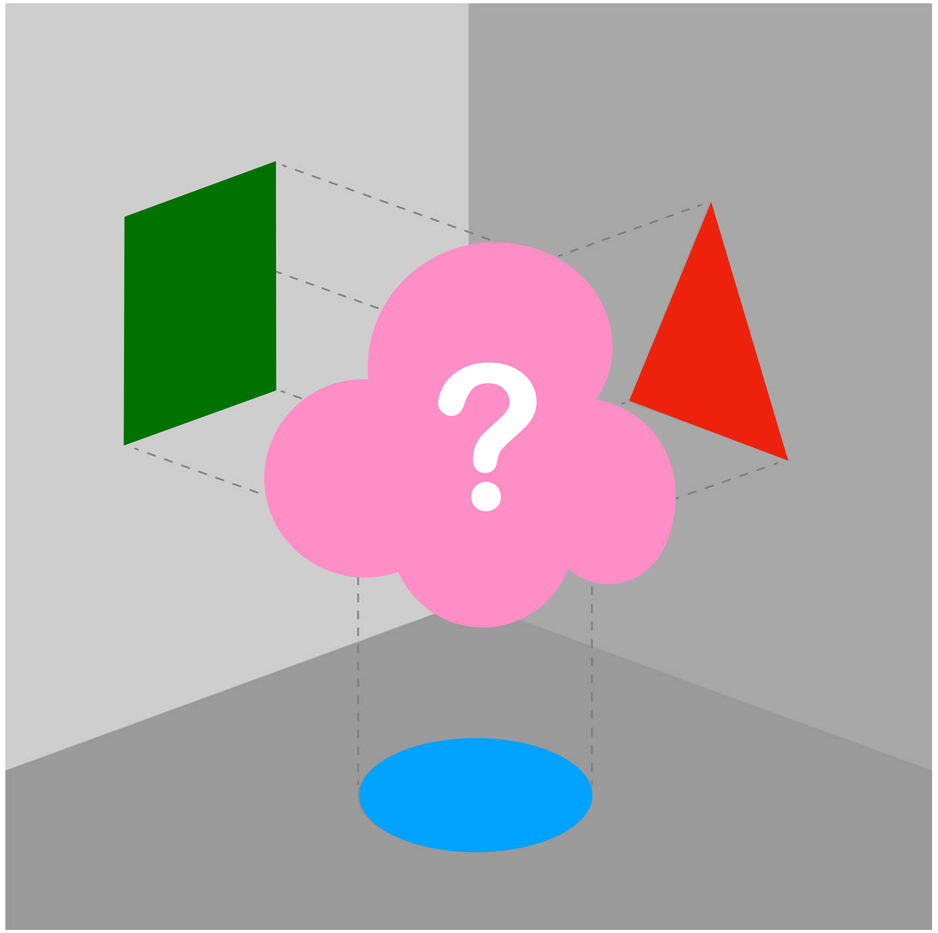
Problem



Each program logic illuminates different aspects but yields an incomplete view

- HL: Correct states on termination
- IL: Some reachable error states

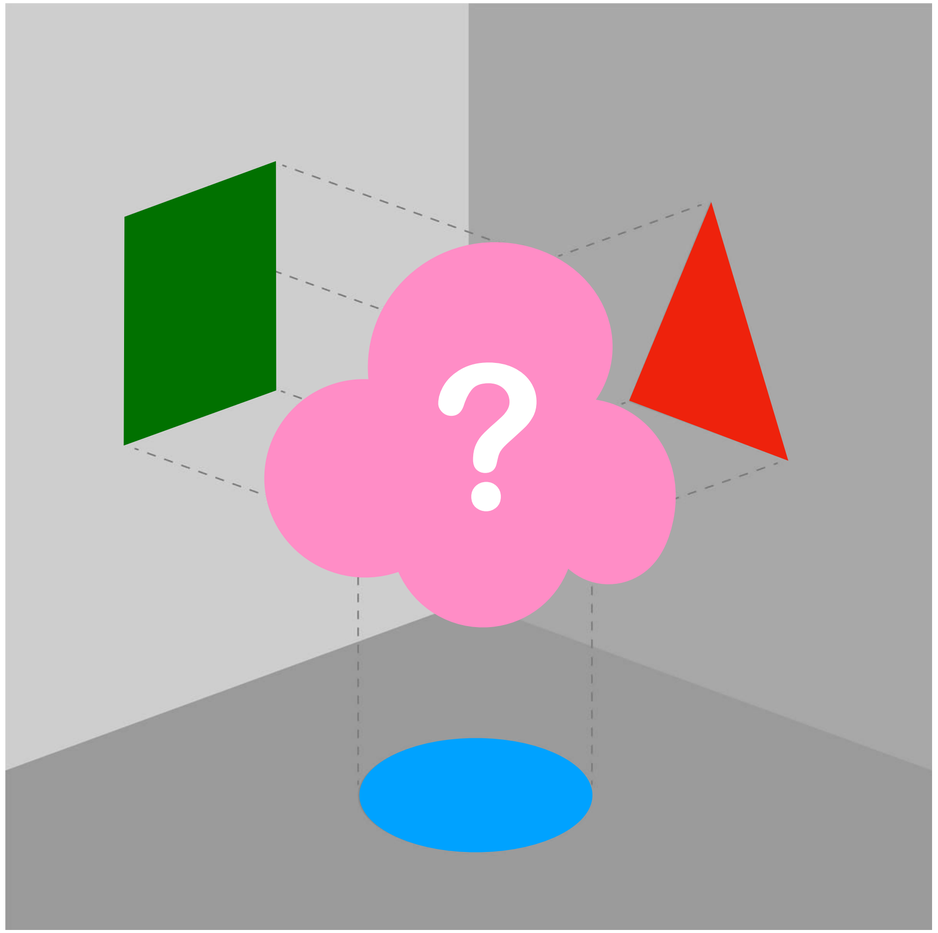
Problem



Each program logic illuminates different aspects but yields an incomplete view

- HL: Correct states on termination
- IL: Some reachable error states
- SIL: Some source of errors states

Problem

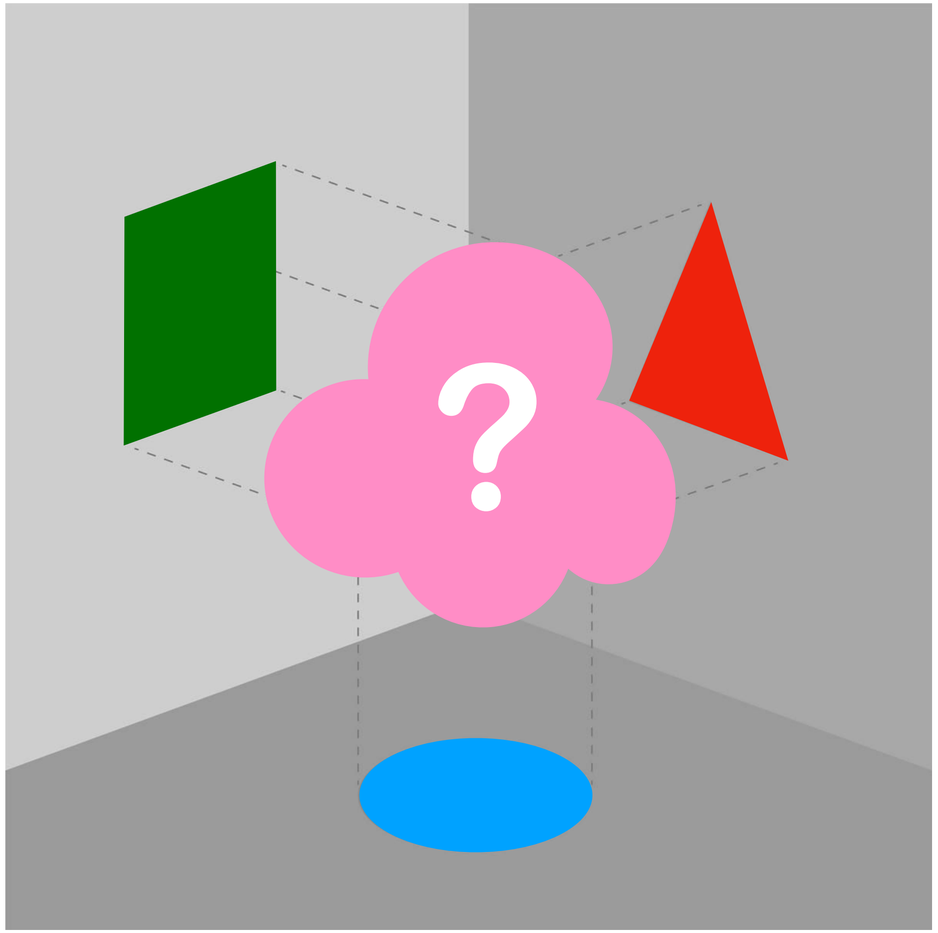


Each program logic illuminates different aspects but yields an incomplete view

- HL: Correct states on termination
- IL: Some reachable error states
- SIL: Some source of errors states

Separate analyses can't reconcile different perspectives into a common view.

Problem



Each program logic illuminates different aspects but yields an incomplete view

- HL: Correct states on termination
- IL: Some reachable error states
- SIL: Some source of errors states

Separate analyses can't reconcile different perspectives into a common view.

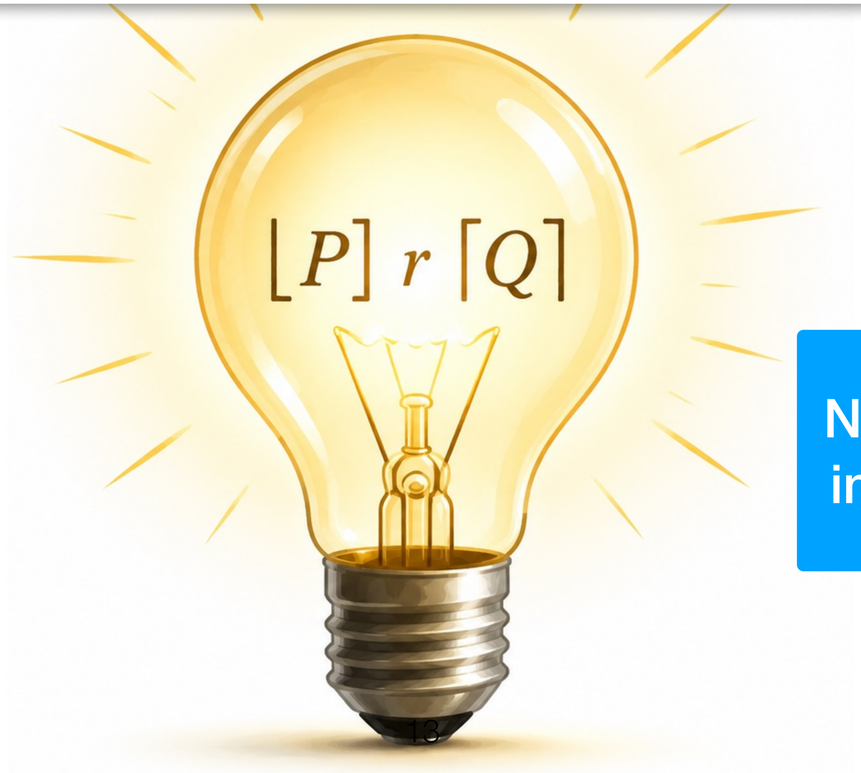
Main objective

- A **novel standard notation** for holistic reasoning
- Encompass **all existing taxonomies**
- Exploit similarities to build a **unique proof system**
- **Combine** multiple analyses to improve precision
- Define a reference, extensible **template**
- Envisioning a **new generation of code summaries**

The idea in a nutshell

Encode approximation by **standard braces**

Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$



Notation is important!

Different logics, different properties

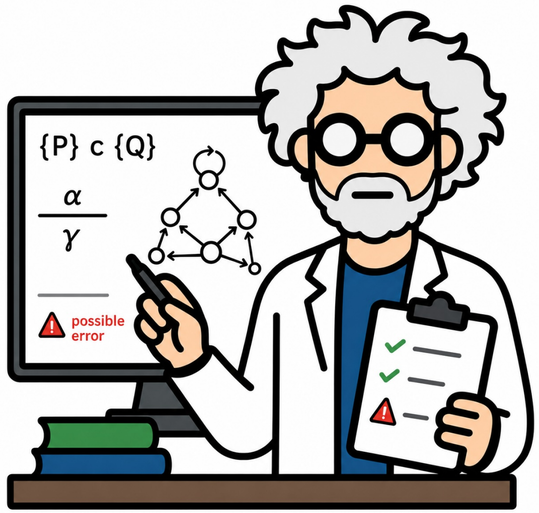
Hi, I'm Bugsy, the "programmer"

Different logics, different properties

Hi, I'm Bugsy, the "programmer"

Different logics, different properties

Hi, I'm Theo, the "analyst"



Hoare logic (HL)

$\{P\}\ r\ \{Q\}$

HL

1

if $\sigma \vDash P$ and

δ is reachable from σ via r,

then $\delta \vDash Q$

Hoare logic (HL)

$\{P\}\ r\ \{Q\}$

HL 1

if $\sigma \vDash P$ and
δ is reachable from σ via r,
then $\delta \vDash Q$

$\{x \geq 0\}$ r $\{x \geq 0\}$

if the pre holds initially

a very large and complicated program

then the post holds upon termination

Hoare logic (HL)

$\{P\}\ r\ \{Q\}$

HL

1

if $\sigma \vDash P$ and
δ is reachable from σ via r,
then $\delta \vDash Q$

notation ambiguity:
are we speaking about partial
or total correctness?

$\{x \geq 0\}$ r $\{x \geq 0\}$

if the pre holds initially

a very large and
complicated program

then the post holds
upon termination

Hoare logic (HL)

$$\{x \geq 0\} \quad r \quad \{x \geq 0\} \; ; \quad y := 1/x$$

if the pre holds initially

a very large and complicated program

then the post holds upon termination

Hoare logic (HL)

can x be 0?

$$\{x \geq 0\} \quad r \quad \{x \geq 0\} \; ; \quad y := 1/x$$

if the pre holds initially

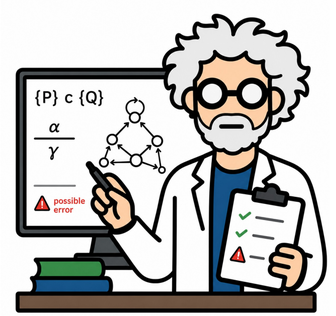
a very large and complicated program

then the post holds upon termination

Hoare logic (HL)

can x be 0?

don't know for sure



$$\{x \geq 0\} \quad r \quad \{x \geq 0\} \; ; \quad y := 1/x$$

if the pre holds initially

a very large and complicated program

then the post holds upon termination

Incorrectness logic (IL, aka FUA, RHL)

$[P]\ r\ [Q]$

IL

2

if $\delta \vDash Q$,
there is some state $\sigma \vDash P$
s.t. δ is reachable from σ via r

$$\{x \geq 0\} \quad r \quad \{x \geq 0\}\ ; \quad y := 1/x$$

a very large and
complicated program

Incorrectness logic (IL, aka FUA, RHL)

$[P]\ r\ [Q]$

IL 2

if $\delta \vDash Q$,
there is some state $\sigma \vDash P$
s.t. δ is reachable from σ via r

$[x \geq 0] \quad r \quad [0 < x \leq 2]; \quad y := 1/x$

a very large and complicated program

any state in the post is reachable

Incorrectness logic (IL, aka FUA, RHL)

can x be 0?

$$[x \geq 0] \quad r \quad [0 < x \leq 2]; \quad y := 1/x$$

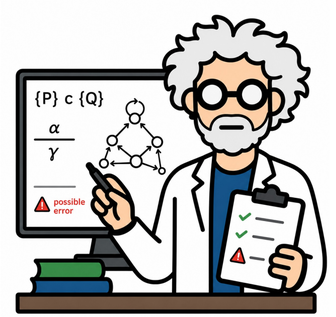
a very large and complicated program

any state in the post is reachable

Incorrectness logic (IL, aka FUA, RHL)

can x be 0?

don't know for sure



$$[x \geq 0] \quad r \quad [0 < x \leq 2]; \quad y := 1/x$$

a very large and complicated program

any state in the post is reachable

Incorrectness logic (IL, aka FUA, RHL)

$$[x \geq 0] \quad r \quad [0 \leq x \leq 2]; \quad y := 1/x$$

a very large and complicated program

any state in the post is reachable

Incorrectness logic (IL, aka FUA, RHL)

can x be 0?

$$[x \geq 0] \quad r \quad [0 \leq x \leq 2]; \quad y := 1/x$$

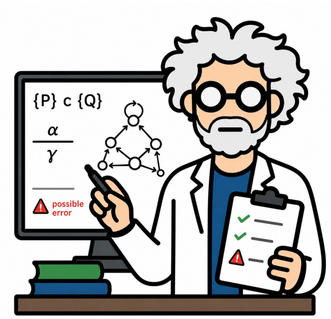
a very large and complicated program

any state in the post is reachable

Incorrectness logic (IL, aka FUA, RHL)

can x be 0?

yes!



$$[x \geq 0] \quad r \quad [0 \leq x \leq 2]; \quad y := 1/x$$

a very large and complicated program

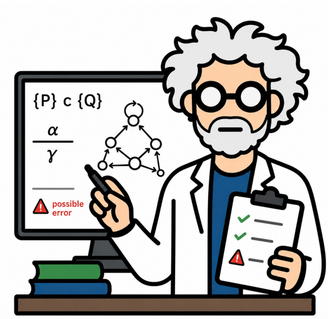
any state in the post is reachable

Incorrectness logic (IL, aka FUA, RHL)

can x be 0?

yes!

for which input?



$$[x \geq 0] \quad r \quad [0 \leq x \leq 2]; \quad y := 1/x$$

a very large and complicated program

any state in the post is reachable

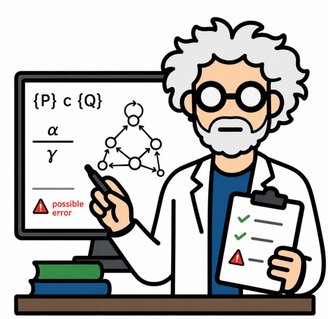
Incorrectness logic (IL, aka FUA, RHL)

can x be 0?

yes!

for which input?

don't know for sure



$$[x \geq 0] \quad r \quad [0 \leq x \leq 2]; \quad y := 1/x$$

a very large and complicated program

any state in the post is reachable

Lisbon triples (aka OL, SIL, BUA)

$\langle P \rangle\ r\ \langle Q \rangle$

SIL

3

if $\sigma \vDash P$,
there is some state $\delta \vDash Q$
s.t. δ is reachable from σ via r

$$[x \geq 0] \quad r \quad [0 < x \leq 2]; \quad y := 1/x$$

a very large and
complicated program

Lisbon triples (aka OL, SIL, BUA)

$\langle P \rangle\ r\ \langle Q \rangle$

SIL

3

if $\sigma \vDash P$,
there is some state $\delta \vDash Q$
s.t. δ is reachable from σ via r

$\langle x \geq 0 \rangle \quad r \quad \langle 0 \leq x \leq 2 \rangle; \quad y := 1/x$

any state in the pre...

a very large and complicated program

...can reach some state in the post

Lisbon triples (aka OL, SIL, BUA)

can x be 0?

$$\langle x \geq 0 \rangle \quad r \quad \langle 0 \leq x \leq 2 \rangle; \quad y := 1/x$$

any state in the pre...

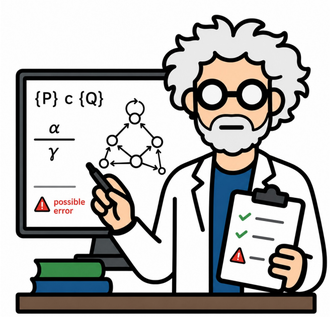
a very large and complicated program

...can reach some state in the post

Lisbon triples (aka OL, SIL, BUA)

can x be 0?

don't know for sure



$$\langle x \geq 0 \rangle \quad r \quad \langle 0 \leq x \leq 2 \rangle; \quad y := 1/x$$

any state in the pre...

a very large and complicated program

...can reach some state in the post

Lisbon triples (aka OL, SIL, BUA)

$$\langle x \geq 0 \rangle \quad r \quad \langle x = 0 \rangle \quad ; \quad y := 1/x$$

any state in the pre...

a very large and complicated program

...can reach some state in the post

Lisbon triples (aka OL, SIL, BUA)

can x be 0?

$$\langle x \geq 0 \rangle \quad r \quad \langle x = 0 \rangle \quad ; \quad y := 1/x$$

any state in the pre...

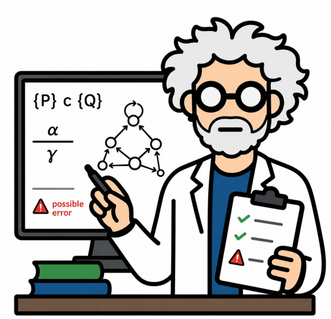
a very large and complicated program

...can reach some state in the post

Lisbon triples (aka OL, SIL, BUA)

can x be 0?

yes!



$$\langle x \geq 0 \rangle \quad r \quad \langle x = 0 \rangle \quad ; \quad y := 1/x$$

any state in the pre...

a very large and complicated program

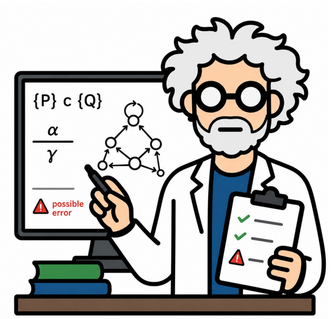
...can reach some state in the post

Lisbon triples (aka OL, SIL, BUA)

can x be 0?

yes!

for which input?



$$\langle x \geq 0 \rangle \quad r \quad \langle x = 0 \rangle \quad ; \quad y := 1/x$$

any state in the pre...

a very large and complicated program

...can reach some state in the post

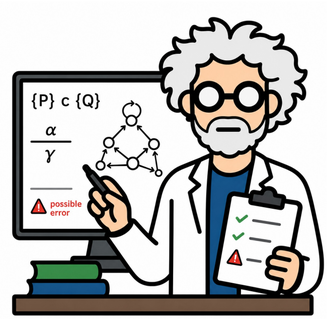
Lisbon triples (aka OL, SIL, BUA)

can x be 0?

yes!

for which input?

for any $x \geq 0$



$$\langle x \geq 0 \rangle \quad r \quad \langle x = 0 \rangle \quad ; \quad y := 1/x$$

any state in the pre...

a very large and complicated program

...can reach some state in the post

Valid triples for different views!

$$\{x \geq 0\} \quad r \quad \{x \geq 0\}$$

$$[x \geq 0] \quad r \quad [0 < x \leq 2]$$

$$[x \geq 0] \quad r \quad [0 \leq x \leq 2]$$

$$\langle x \geq 0 \rangle \quad r \quad \langle 0 \leq x \leq 2 \rangle$$

$$\langle x \geq 0 \rangle \quad r \quad \langle x = 0 \rangle$$

An observation

HL

1

$\{P\}\ r\ \{Q\}$

if $\sigma \vDash P$ and
δ is reachable from σ via r,
then $\delta \vDash Q$

why are pre and post surrounded with the same braces if their roles are pretty different?

IL

2

$[P]\ r\ [Q]$

if $\delta \vDash Q$,
there is some state $\sigma \vDash P$
s.t. δ is reachable from σ via r

SIL

3

$\langle P \rangle\ r\ \langle Q \rangle$

if $\sigma \vDash P$,
there is some state $\delta \vDash Q$
s.t. δ is reachable from σ via r

An observation

HL

1

$\{P\}\ r\ \{Q\}$

if $\sigma \vDash P$ and
δ is reachable from σ via r,
then $\delta \vDash Q$

why are pre and post surrounded with the same braces if their roles are pretty different?

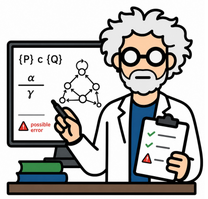
IL

2

$[P]\ r\ [Q]$

if $\delta \vDash Q$,
there is some state $\sigma \vDash P$
s.t. δ is reachable from σ via r

don't know for sure



SIL

3

$\langle P\rangle\ r\ \langle Q\rangle$

if $\sigma \vDash P$,
there is some state $\delta \vDash Q$
s.t. δ is reachable from σ via r

Background

Collecting semantics

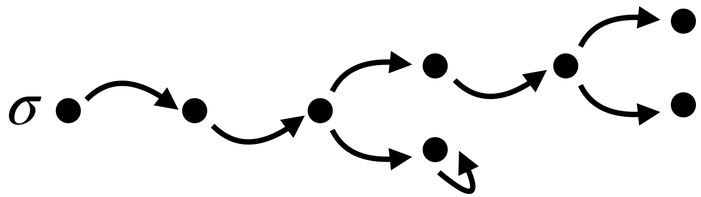
$\mathsf{Reg} \ni r ::= c \mid r_1; r_2 \mid r_1 + r_2 \mid r^*$

$[\![r]\!] : \Sigma \to \wp(\Sigma)$

Collecting semantics

Reg $\ni$ r ::= c | $r_1; r_2$ | $r_1 + r_2$ | r*

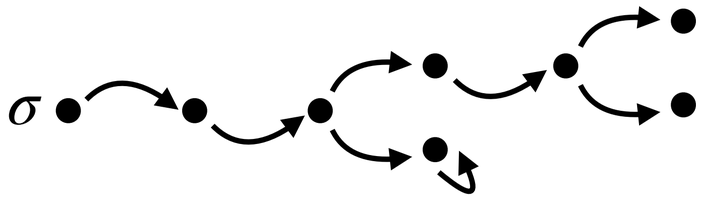
$[\![r]\!] : \Sigma \to \wp(\Sigma)$

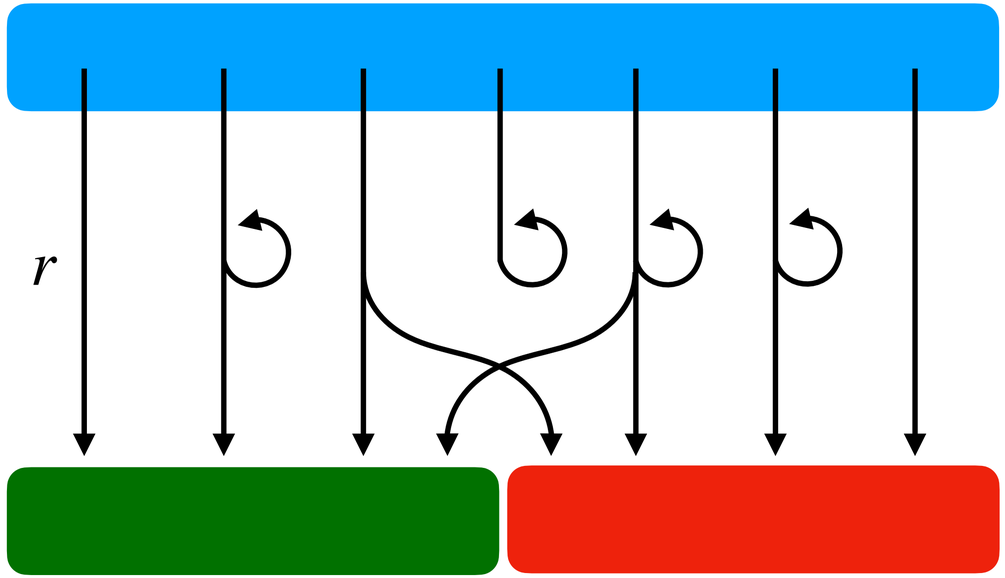


Collecting semantics

Reg $\ni$ r ::= c | $r_1; r_2$ | $r_1 + r_2$ | r*

$[\![r]\!] : \Sigma \to \wp(\Sigma)$

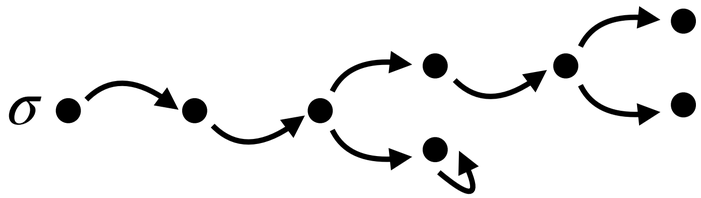




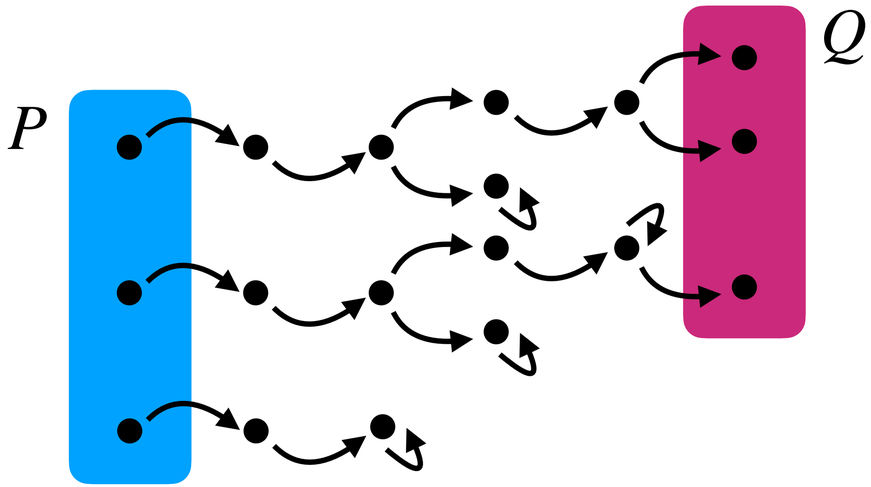
Collecting semantics

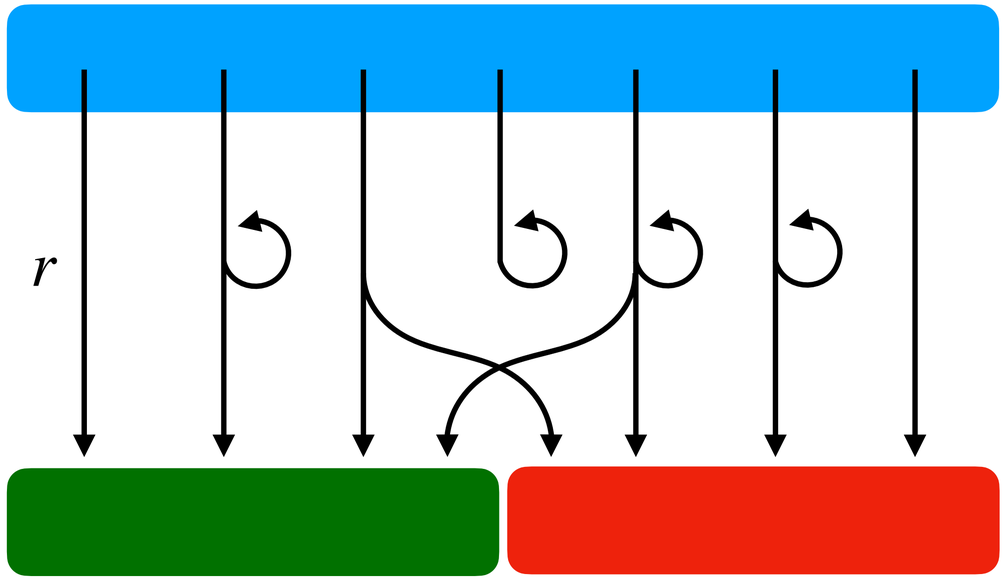
Reg ∋ r ::= c | $r_1; r_2$ | $r_1 + r_2$ | r*

$[\![r]\!] : \Sigma \to \wp(\Sigma)$



$[\![r]\!] : \wp(\Sigma) \to \wp(\Sigma)$





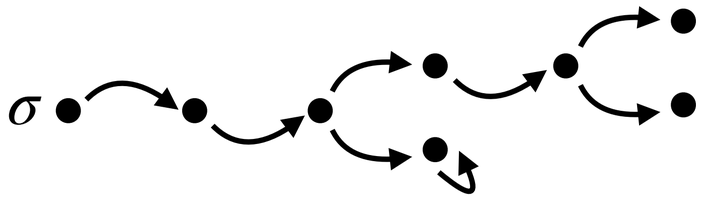
$$asp(P, r) = [\![r]\!]P = Q$$

notation: [POPL25] Verscht and Kaminski

Backward semantics

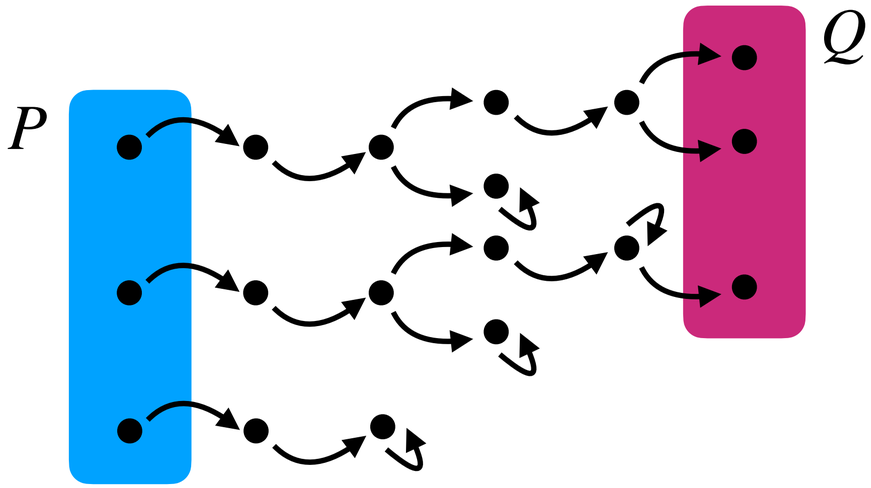
$\mathsf{Reg} \ni \mathsf{r} ::= \mathsf{c} \mid \mathsf{r}_1; \mathsf{r}_2 \mid \mathsf{r}_1 + \mathsf{r}_2 \mid \mathsf{r}^*$

$[\![\mathsf{r}]\!] : \Sigma \to \wp(\Sigma)$



$[\![\mathsf{r}]\!] : \wp(\Sigma) \to \wp(\Sigma)$

$[\![\overleftarrow{\mathsf{r}}]\!] : \wp(\Sigma) \to \wp(\Sigma)$



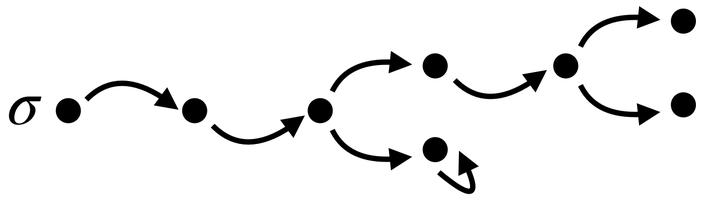
$$asp(P, r) = [\![\mathsf{r}]\!]P = Q$$

notation: [POPL25] Verscht and Kaminski

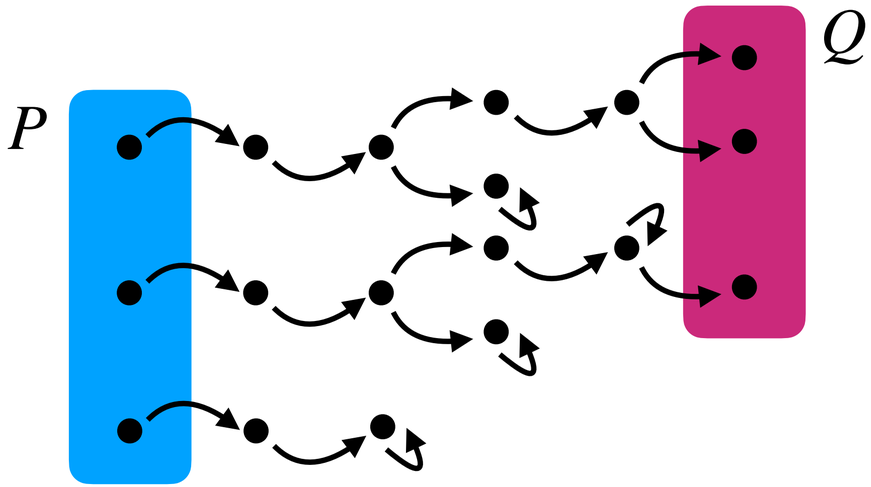
Backward semantics

Reg ∋ r ::= c | $r_1; r_2$ | $r_1 + r_2$ | r*

$[\![r]\!] : \Sigma \to \wp(\Sigma)$



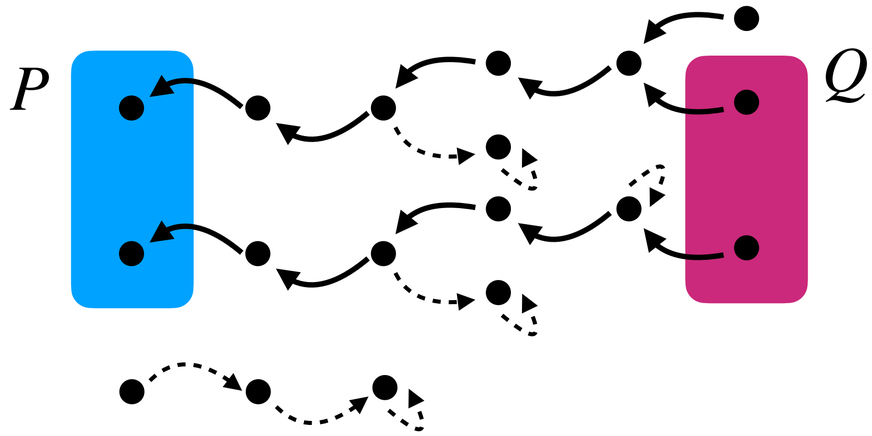
$[\![r]\!] : \wp(\Sigma) \to \wp(\Sigma)$



$$asp(P, r) = [\![r]\!]P = Q$$

notation: [POPL25] Verscht and Kaminski

$[\![\overleftarrow{r}]\!] : \wp(\Sigma) \to \wp(\Sigma)$



$$awp(Q, r) = P$$

notation: [POPL25] Verscht and Kaminski

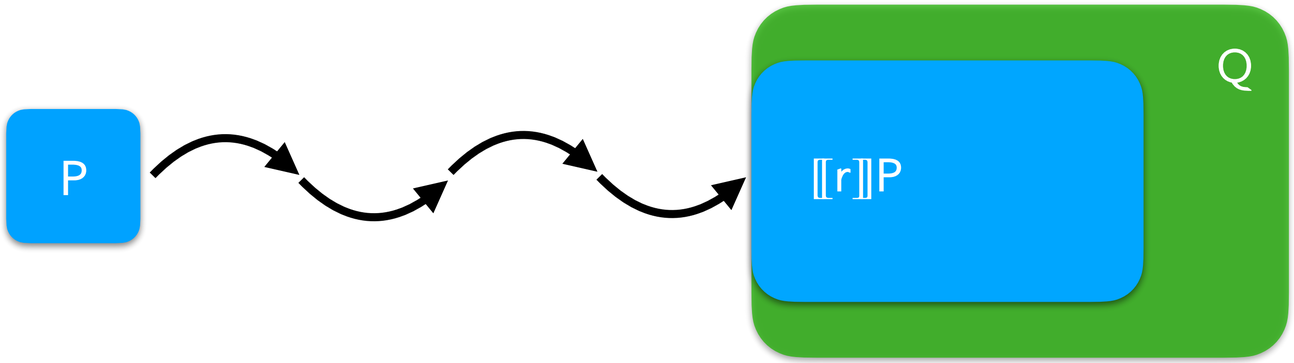
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	
Under		



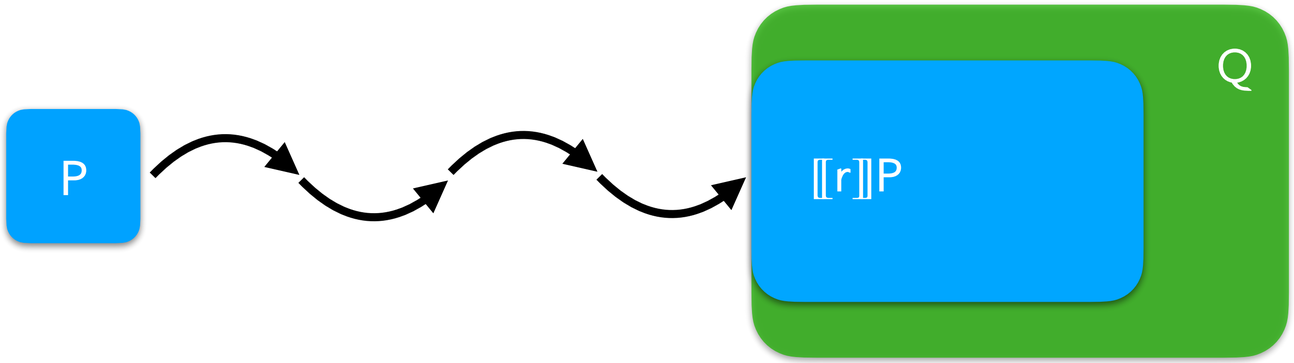
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	
Under		



🐞 $\notin Q \Rightarrow$ 🐞 $\notin [\![r]\!]P$

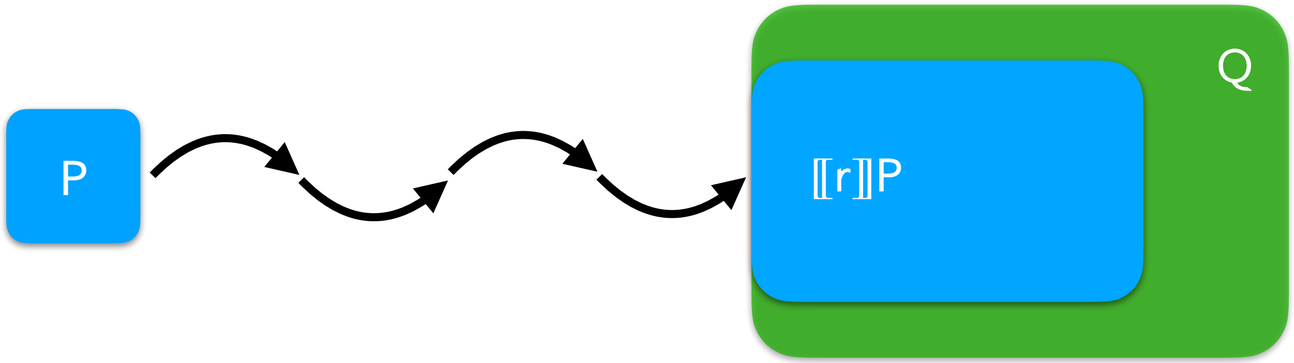
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	
Under		



🐞 $\notin Q \Rightarrow$ 🐞 $\notin [\![r]\!]P$

🐞 $\in Q \Rightarrow$ May be a false positive

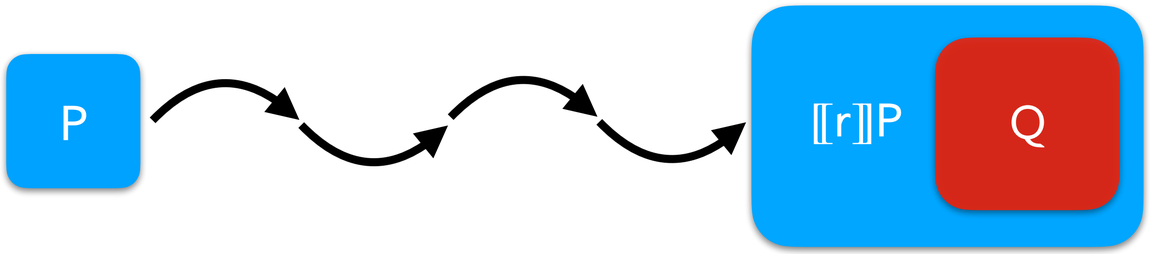
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	
Under	[POPL20] Incorrectness Logic (IL) $[P]\ r\ [Q] \iff Q \subseteq [\![r]\!]P$ Every (error) state in Q is reachable from some P	



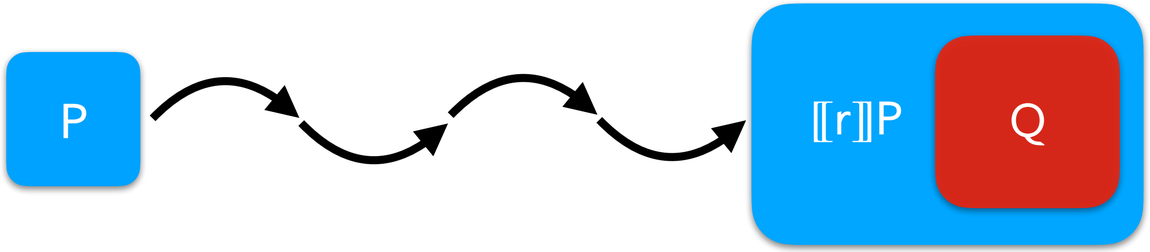
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	
Under	[POPL20] Incorrectness Logic (IL) $[P]\ r\ [Q] \iff Q \subseteq [\![r]\!]P$ Every (error) state in Q is reachable from some P	



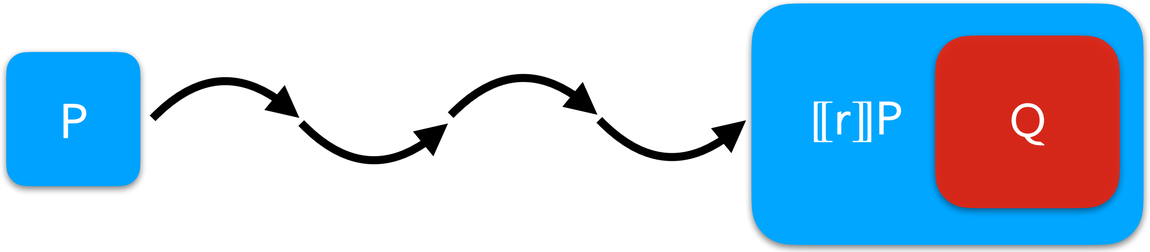
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	
Under	[POPL20] Incorrectness Logic (IL) $[P]\ r\ [Q] \iff Q \subseteq [\![r]\!]P$ Every (error) state in Q is reachable from some P	



🐞 $\in Q \Rightarrow$ 🐞 $\in [\![r]\!]P$

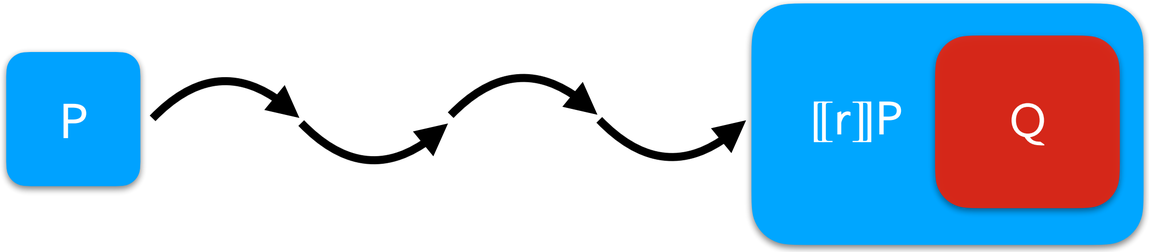
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	
Under	[POPL20] Incorrectness Logic (IL) $[P]\ r\ [Q] \iff Q \subseteq [\![r]\!]P$ Every (error) state in Q is reachable from some P	



🐞 $\in Q \Rightarrow$ 🐞 $\in [\![r]\!]P$

🐞 $\notin Q \Rightarrow$ May be a false negative

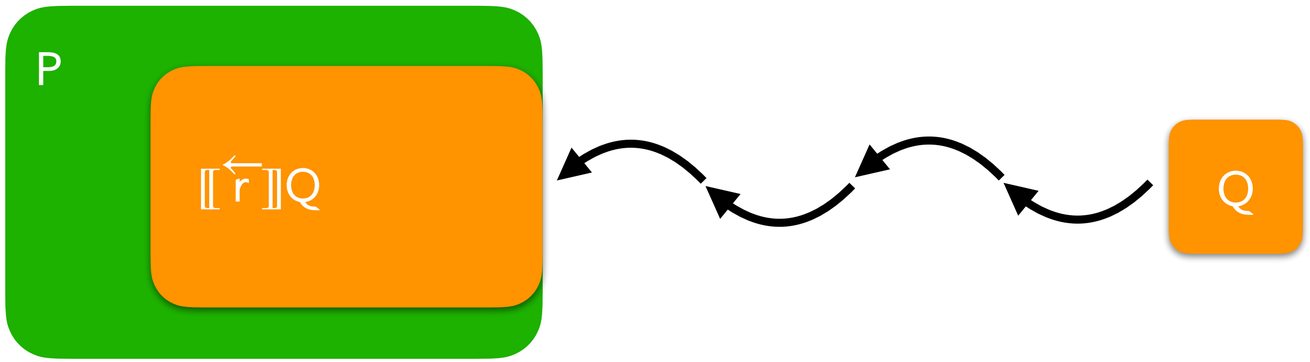
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	[VMCAI13] Necessary Conditions (NC) $(P)\ r\ (Q) \iff [\![\overleftarrow{r}]\!]Q \subseteq P$ Q is unreachable from states outside P
Under	[POPL20] Incorrectness Logic (IL) $[P]\ r\ [Q] \iff Q \subseteq [\![r]\!]P$ Every (error) state in Q is reachable from some P	



If our input is not in P,
then we cannot reach Q

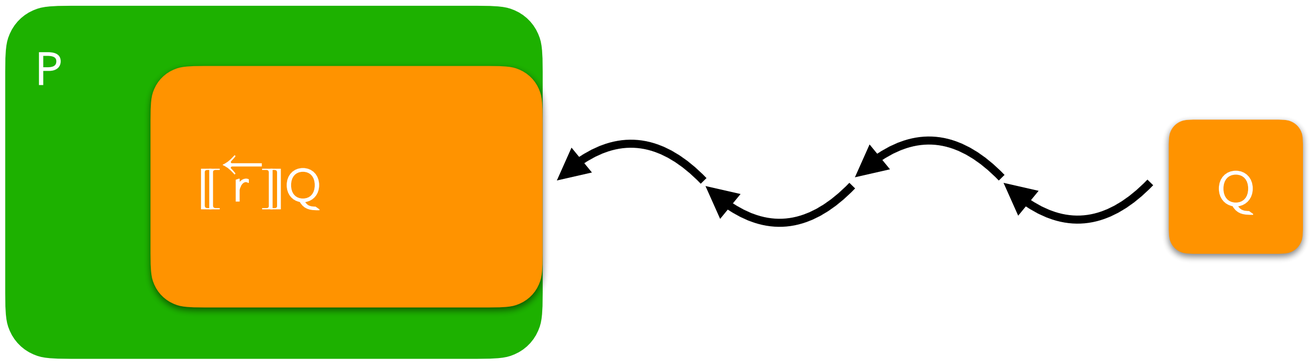
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	[VMCAI13] Necessary Conditions (NC) $(P)\ r\ (Q) \iff [\![\overleftarrow{r}]\!]Q \subseteq P$ Q is unreachable from states outside P
Under	[POPL20] Incorrectness Logic (IL) $[P]\ r\ [Q] \iff Q \subseteq [\![r]\!]P$ Every (error) state in Q is reachable from some P	



If our input is not in P,
then we cannot reach Q

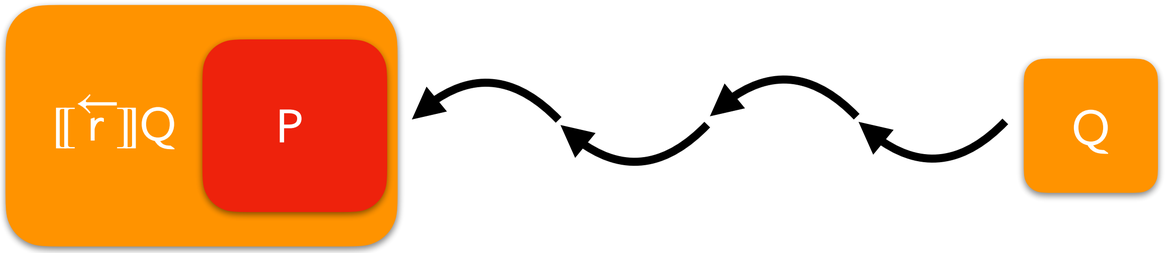
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	[VMCAI13] Necessary Conditions (NC) $(P)\ r\ (Q) \iff [\![\overleftarrow{r}]\!]Q \subseteq P$ Q is unreachable from states outside P
Under	[POPL20] Incorrectness Logic (IL) $[P]\ r\ [Q] \iff Q \subseteq [\![r]\!]P$ Every (error) state in Q is reachable from some P	[OOPSLA25] Sufficient Incorrectness Logic (SIL) $\langle P\rangle\ r\ \langle Q\rangle \iff P \subseteq [\![\overleftarrow{r}]\!]Q$ From every state in P some (error) state in Q is reachable



Any state in P can reach some state in Q

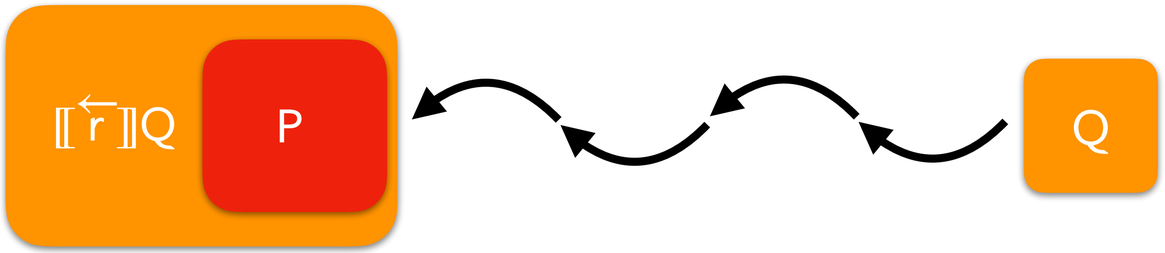
Background: program logics

[OOPSLA25] Ascari, Bruni, Gori, Logozzo

Over vs Under

Forward vs Backward

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff [\![r]\!]P \subseteq Q$ Q error-free ⇒ no errors are possible	[VMCAI13] Necessary Conditions (NC) $(P)\ r\ (Q) \iff [\![\overleftarrow{r}]\!]Q \subseteq P$ Q is unreachable from states outside P
Under	[POPL20] Incorrectness Logic (IL) $[P]\ r\ [Q] \iff Q \subseteq [\![r]\!]P$ Every (error) state in Q is reachable from some P	[OOPSLA25] Sufficient Incorrectness Logic (SIL) $\langle P\rangle\ r\ \langle Q\rangle \iff P \subseteq [\![\overleftarrow{r}]\!]Q$ From every state in P some (error) state in Q is reachable



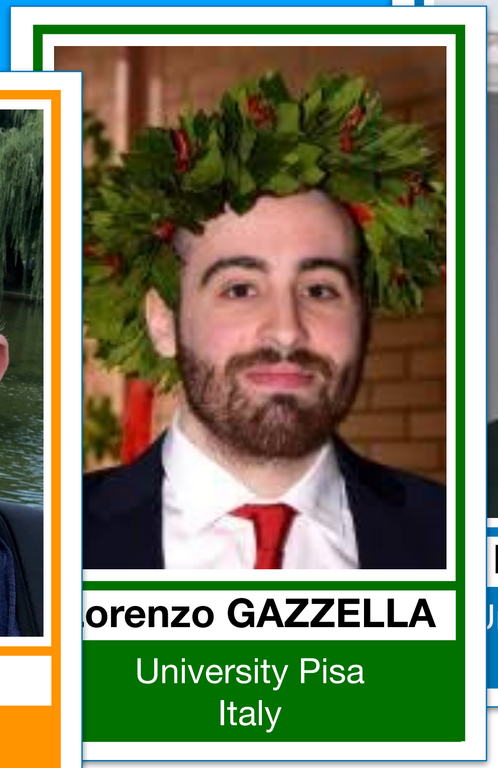
Any state in P can reach some state in Q

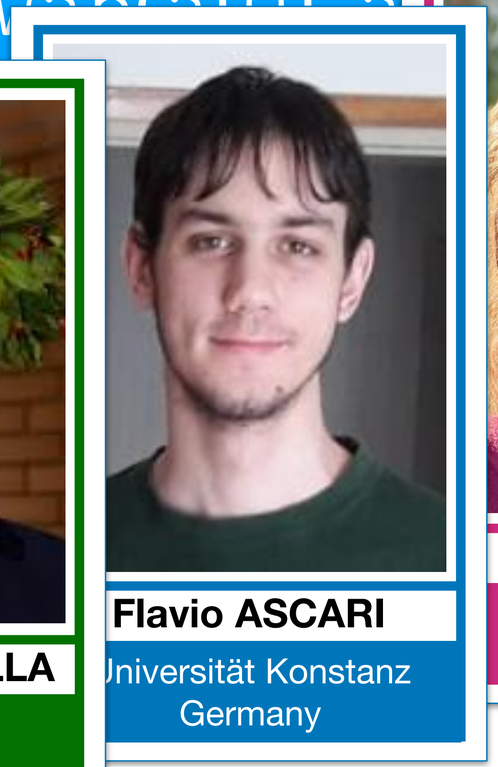
A sample program: shuffle

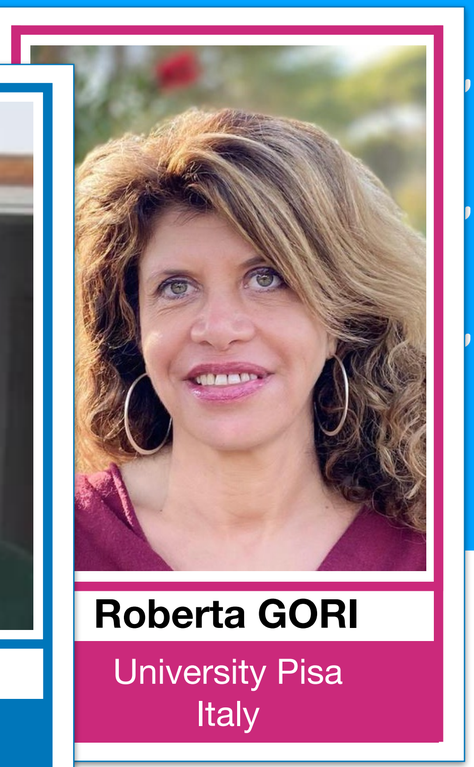
```
shuffle ≜ a := copy(b);
          swap(a[0], a[nondet(0, a.len − 1)])
          swap(a[1], a[nondet(0, a.len − 1)])
          swap(a[2], a[nondet(0, a.len − 1)])
```

A sample program: shuffle

```
shuffle ≜ a := copy(b);
          swap(a[0], a[rand(0, a.len − 1)])
          swap(a[1], a[rand(1, a.len − 1)])
          swap(a[2], a[rand(2, a.len − 1)])
```



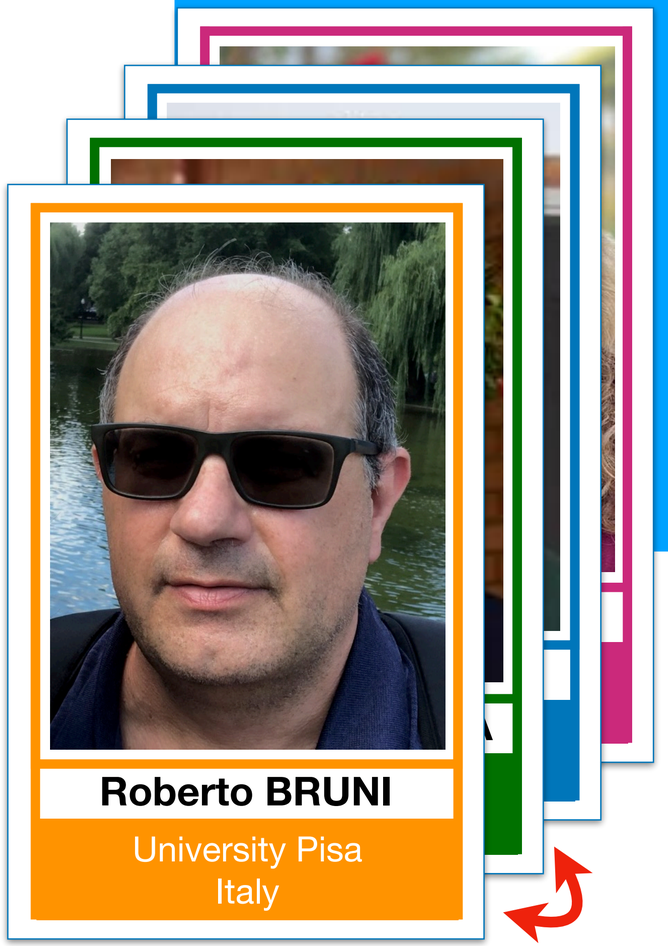




A sample program: shuffle

```
shuffle ≜ a := copy(b);
          swap(a[0], a[nondet(0, a.len − 1)])
          swap(a[1], a[nondet(0, a.len − 1)])
          swap(a[2], a[nondet(0, a.len − 1)])
```

1



Rotate an array of one position

A sample program: shuffle

```
shuffle ≜ a := copy(b);
          swap(a[0], a[nondet(0, a.len − 1)])
          swap(a[1], a[nondet(0, a.len − 1)])
          swap(a[2], a[nondet(0, a.len − 1)])
```

2

Rotate an array of one position

A sample program: shuffle

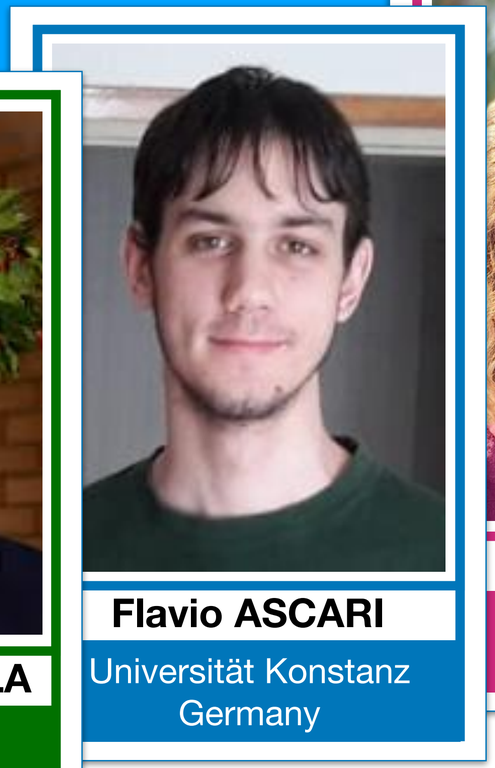
```
shuffle ≜ a := copy(b);
          swap(a[0], a[nondet(0, a.len − 1)])
          swap(a[1], a[nondet(0, a.len − 1)])
          swap(a[2], a[nondet(0, a.len − 1)])
```

Rotate an array of one position

A sample program: shuffle

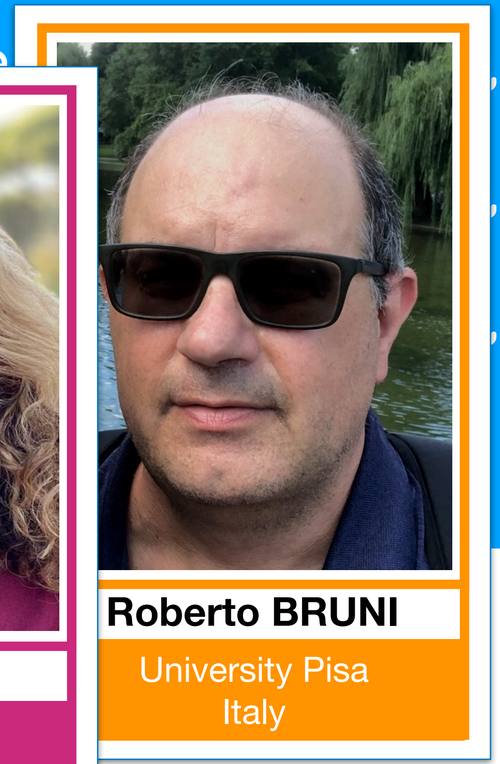
```
shuffle ≜ a := copy(b);
          swap(a[0], a[rand(0, b.len − 1)])
          swap(a[1], a[rand(1, b.len − 1)])
          swap(a[2], a[rand(2, b.len − 1)])
```

Lorenzo GAZZELLA
University Pisa
Italy



Flavio ASCARI
Universität Konstanz
Germany

Roberta GORI
University Pisa
Italy



Roberto BRUNI
University Pisa
Italy

`b.len − 1` **swaps**

A sample program: shuffle

```
shuffle ≜ a := copy(b);
          m := 0;
          swap(a[m++], a[nondet(0, a.len − 1)])^k
```

scan index m

use m and update m

repeat k times

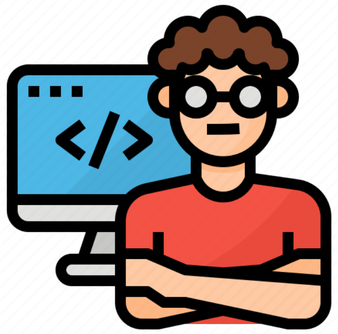
A sample program: shuffle

```
shuffle ≜ a := copy(b);
          m := 0;
          swap(a[m++], a[nondet(0, a.len − 1)])^k
```

scan index m

use m and update m

repeat k times



➡ Can we reach a state that is not a permutation of b?

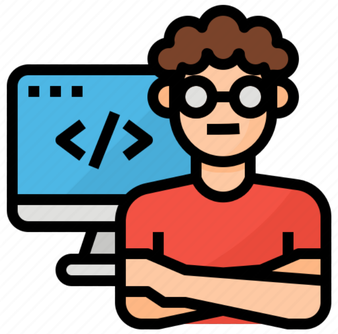
A sample program: shuffle

```
shuffle ≜ a := copy(b);
          m := 0;
          swap(a[m++], a[nondet(0, a.len − 1)])^k
```

scan index m

use m and update m

repeat k times



➡ Can we reach a state that is not a permutation of b?

➡ Can we reach errors?

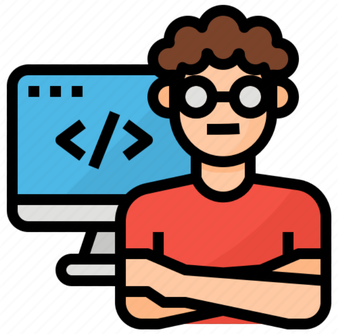
A sample program: shuffle

```
shuffle ≜ a := copy(b);
          m := 0;
          swap(a[m++], a[nondet(0, a.len − 1)])^k
```

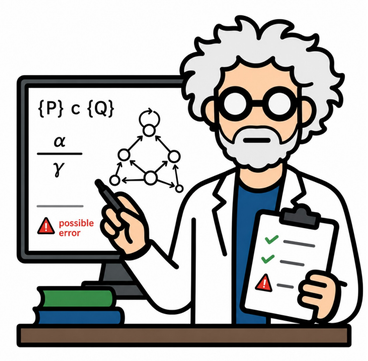
scan index m

use m and update m

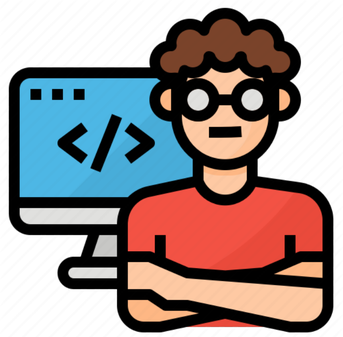
repeat k times



- Can we reach a state that is not a permutation of b?
- Can we reach errors?
- Which sufficient condition for reaching the reversal $b^\dagger$ of b?

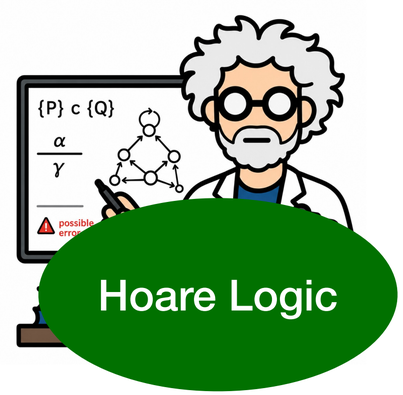


Examples



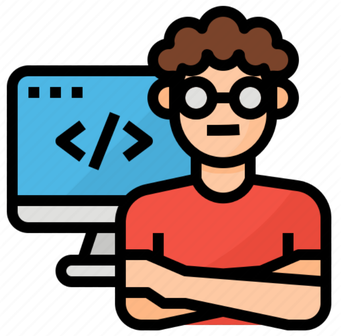
Using program shuffle:

- Can we reach a state that is not a permutation of b?
- Can we reach errors?
- Which sufficient condition for reaching the reversal $b^\dagger$ of b?



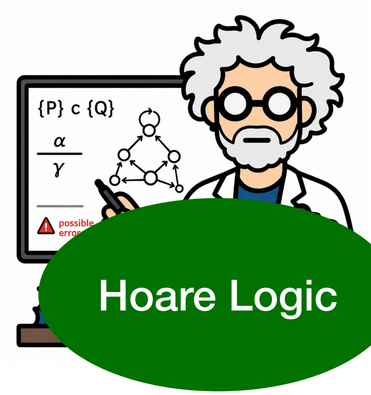
Examples

$\{k < b.len\}$ `shuffle` $\{a \in perm(b)\}$



Using program shuffle:

- Can we reach a state that is not a permutation of b?
- Can we reach errors?
- Which sufficient condition for reaching the reversal $b^{\dagger}$ of b?



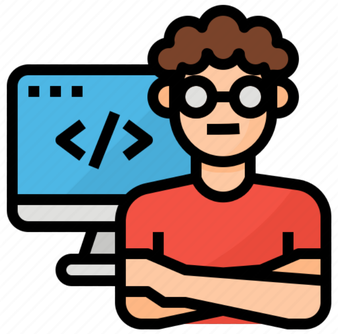
Examples

$\{k < b.len\}$ `shuffle` $\{a \in perm(b)\}$

Incorrectness Logic

$[k > b.len]$ `shuffle` $[er : k > m = b.len \wedge a \in perm(b)]$

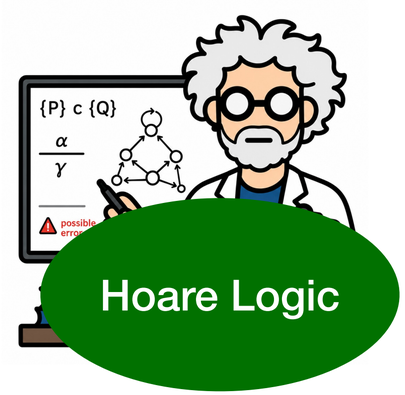
Out of bound



Using program shuffle:

- Can we reach a state that is not a permutation of b?
- Can we reach errors?
- Which sufficient condition for reaching the reversal $b^{\dagger}$ of b?

Examples



Hoare Logic

$\{k < b.len\}$ shuffle $\{a \in perm(b)\}$

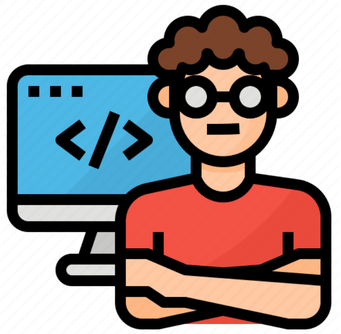
Incorrectness Logic

$[k > b.len]$ shuffle $[er : k > m = b.len \wedge a \in perm(b)]$

Out of bound

Sufficient Incorrectness Logic

$\langle k = b.len/2 \rangle$ shuffle $\langle a = b^{\dagger} \rangle$



Using program shuffle:

- Can we reach a state that is not a permutation of b?
- Can we reach errors?
- Which sufficient condition for reaching the reversal $b^{\dagger}$ of b?

A Novel Notation

A novel notation

Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$

A novel notation

Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff (P)\ r\ \lceil Q \rceil$	
Under		

A novel notation

Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff (P)\ r\ \lceil Q \rceil$	[VMCAI13] Necessary Conditions (NC) $(P)\ r\ (Q) \iff \lceil P \rceil\ r\ (Q)$
Under		

A novel notation

Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff (P)\ r\ \lceil Q \rceil$	[VMCAI13] Necessary Conditions (NC) $(P)\ r\ (Q) \iff \lceil P \rceil\ r\ (Q)$
Under	$[P]\ r\ [Q] \iff (P)\ r\ \lfloor Q \rfloor$ [POPL20] Incorrectness Logic (IL)	

A novel notation

Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff (P)\ r\ \lceil Q \rceil$	[VMCAI13] Necessary Conditions (NC) $(P)\ r\ (Q) \iff \lceil P \rceil\ r\ (Q)$
Under	$[P]\ r\ [Q] \iff (P)\ r\ \lfloor Q \rfloor$ [POPL20] Incorrectness Logic (IL)	$\langle P \rangle\ r\ \langle Q \rangle \iff \lfloor P \rfloor\ r\ (Q)$ [OOPSLA25] Sufficient Incorrectness Logic (SIL)

A novel notation

Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff (P)\ r\ \lceil Q \rceil$	[VMCAI13] Necessary Conditions (NC) $(P)\ r\ (Q) \iff \lceil P \rceil\ r\ (Q)$
Exact	[ECOOP23] ESL $(P)\ r\ \lvert Q \rvert$	$\lvert P \rvert\ r\ (Q)$
Under	$[P]\ r\ [Q] \iff (P)\ r\ \lfloor Q \rfloor$ [POPL20] Incorrectness Logic (IL)	$\langle P \rangle\ r\ \langle Q \rangle \iff \lfloor P \rfloor\ r\ (Q)$ [OOPSLA25] Sufficient Incorrectness Logic (SIL)

A novel notation

Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$

	Forward	**Backward**
Over	Hoare Logic (HL) $\{P\}\ r\ \{Q\} \iff (P)\ r\ \lceil Q \rceil$	[VMCAI13] Necessary Conditions (NC) $(P)\ r\ (Q) \iff \lceil P \rceil\ r\ (Q)$
Exact	[ECOOP23] ESL $(P)\ r\ \lvert Q \rvert$	$\lvert P \rvert\ r\ (Q)$
Under	$[P]\ r\ [Q] \iff (P)\ r\ \lfloor Q \rfloor$ [POPL20] Incorrectness Logic (IL)	$\langle P \rangle\ r\ \langle Q \rangle \iff \lfloor P \rfloor\ r\ (Q)$ [OOPSLA25] Sufficient Incorrectness Logic (SIL)

All triples are ASYMMETRIC: only one side is constrained!

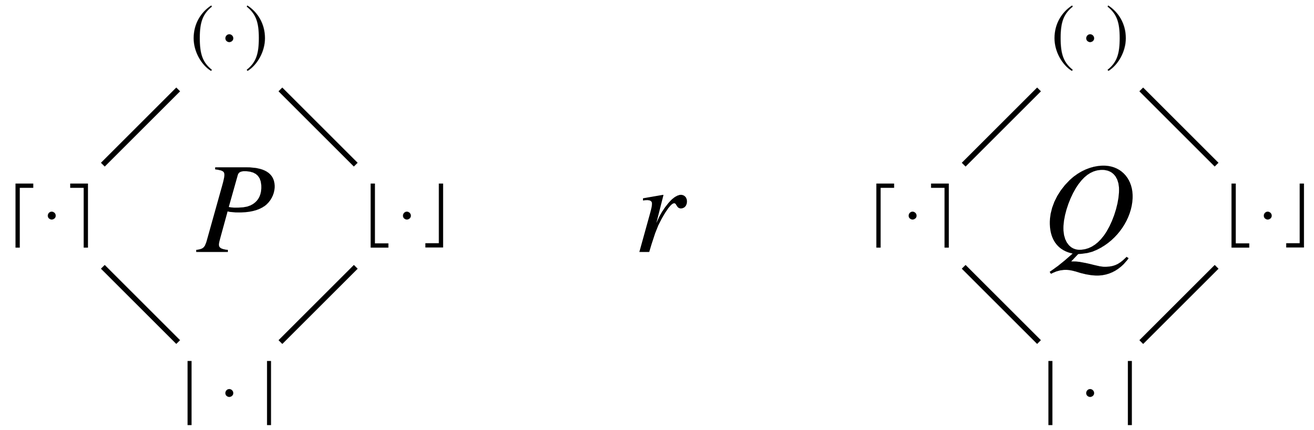
The hardest things to find are sometimes those hidden in plain sight.

16 possible analyses

$(\!| P |\!)_\alpha \; r \; (\!| Q |\!)_\beta$

metabraces

- Q is a β-approximation of $[\![\mathsf{r}]\!]P$
- P is a α-approximation of $[\![\overleftarrow{\mathsf{r}}]\!]Q$



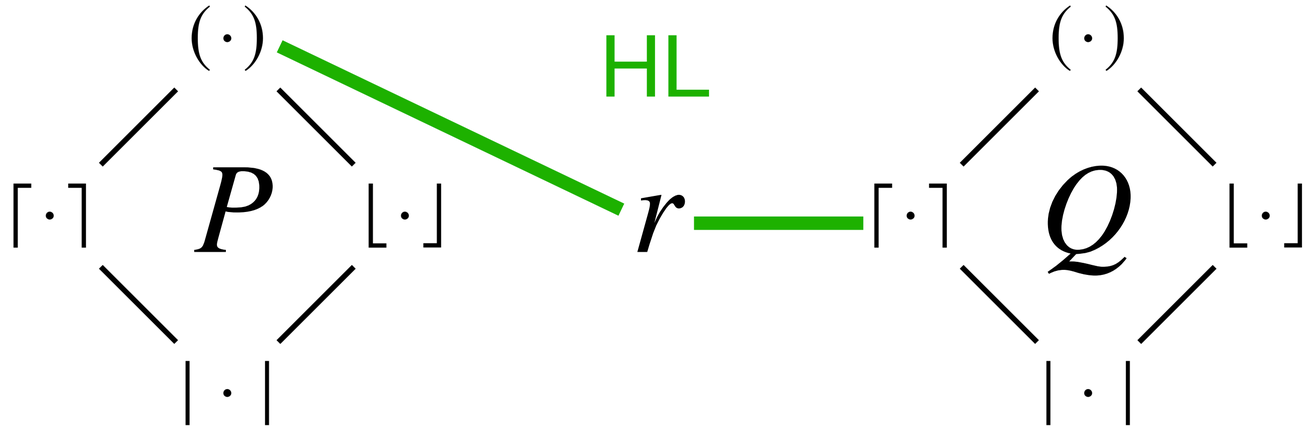
If approximation is a first-class concept, every combination becomes meaningful

16 possible analyses

$$\langle\!|\, P \,|\!\rangle_\alpha \; r \; \langle\!|\, Q \,|\!\rangle_\beta$$

metabraces

- Q is a β-approximation of $[\![\mathrm{r}]\!]P$
- P is a α-approximation of $[\![\overleftarrow{\mathrm{r}}]\!]Q$

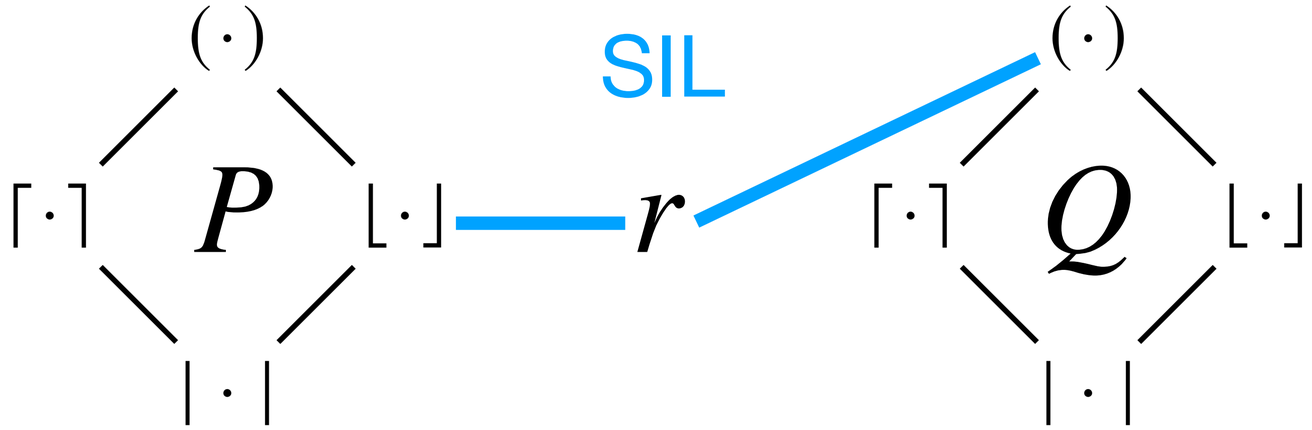


If approximation is a first-class concept, every combination becomes meaningful

16 possible analyses

$(\!| P |\!)_\alpha \; r \; (\!| Q |\!)_\beta$

- Q is a β-approximation of $[\![\mathsf{r}]\!]P$
- P is a α-approximation of $[\![\overleftarrow{\mathsf{r}}]\!]Q$

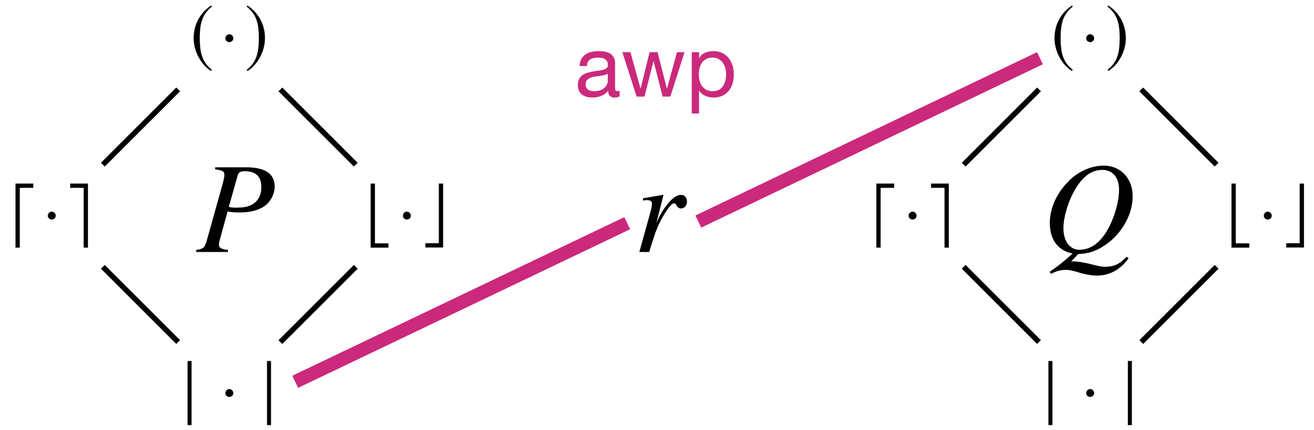


If approximation is a first-class concept, every combination becomes meaningful

16 possible analyses

$$(\!| P |\!)_\alpha \; r \; (\!| Q |\!)_\beta$$

- Q is a β-approximation of $[\![\mathsf{r}]\!]P$
- P is a α-approximation of $[\![\overleftarrow{\mathsf{r}}]\!]Q$

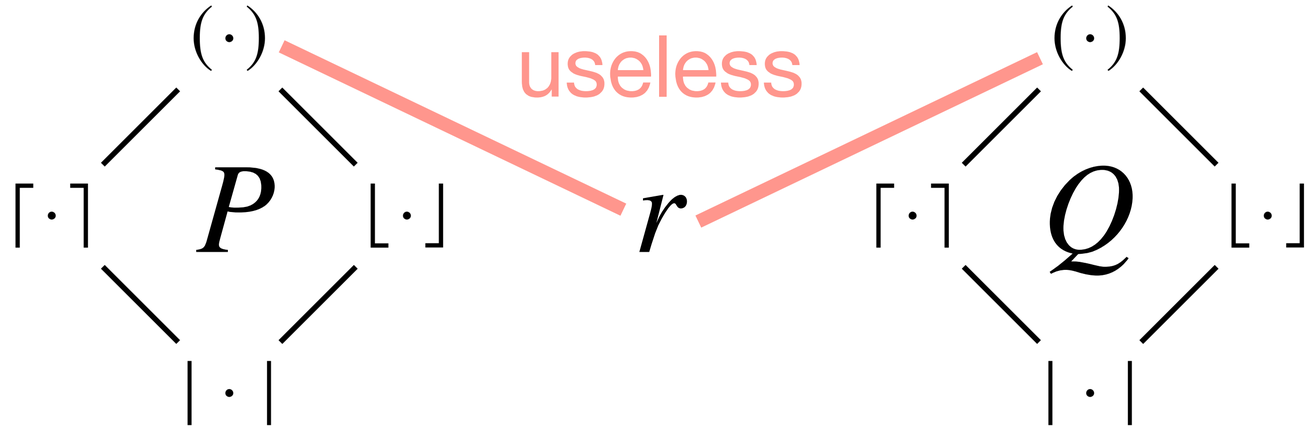


If approximation is a first-class concept, every combination becomes meaningful

16 possible analyses (minus 1)

$$(\!| P |\!)_\alpha \; r \; (\!| Q |\!)_\beta$$

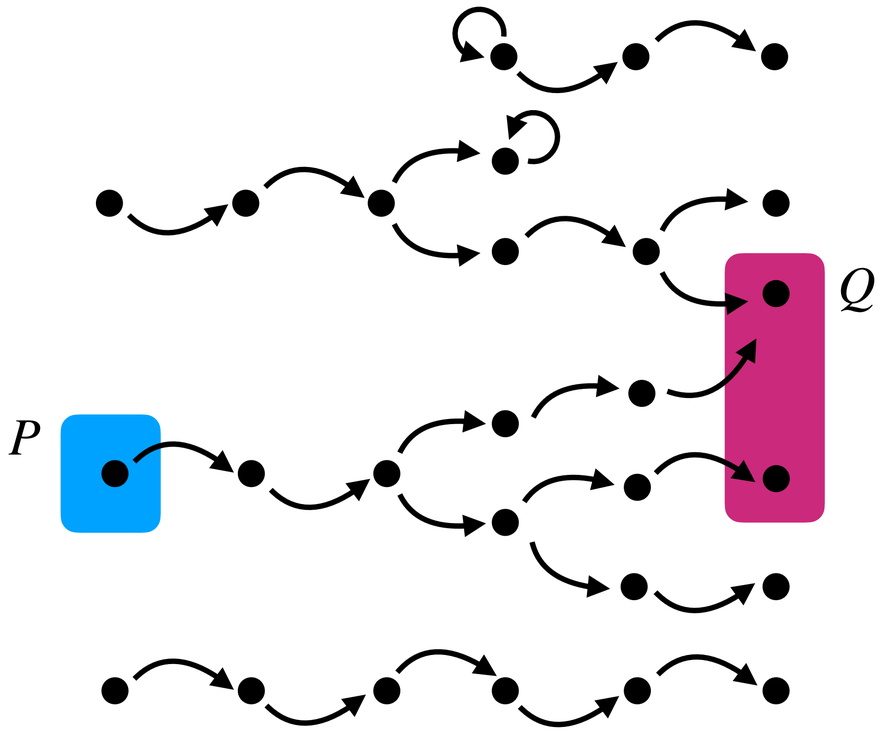
- Q is a β-approximation of $[\![\mathsf{r}]\!]P$
- P is a α-approximation of $[\![\overleftarrow{\mathsf{r}}]\!]Q$



If approximation is a first-class concept, most combinations become meaningful

Under-under

Also studied in [OOPSLA24] UnTer and [POPL26] U-Turn

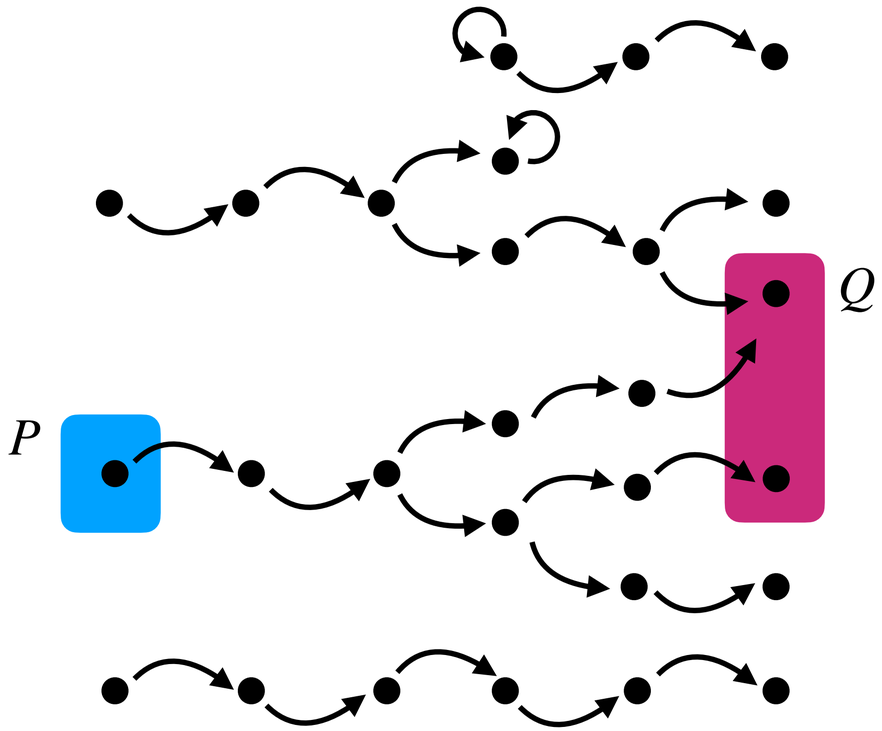


$$\lfloor P \rfloor\ r\ \lfloor Q \rfloor \quad \equiv \quad Q \subseteq [\![\mathsf{r}]\!]P \quad \wedge \quad P \subseteq [\![\overleftarrow{\mathsf{r}}]\!]Q$$

- ➡ Every state in Q is reachable
- ➡ Every state in P is guaranteed to reach a state in Q

Under-under

Also studied in [OOPSLA24] UnTer and [POPL26] U-Turn

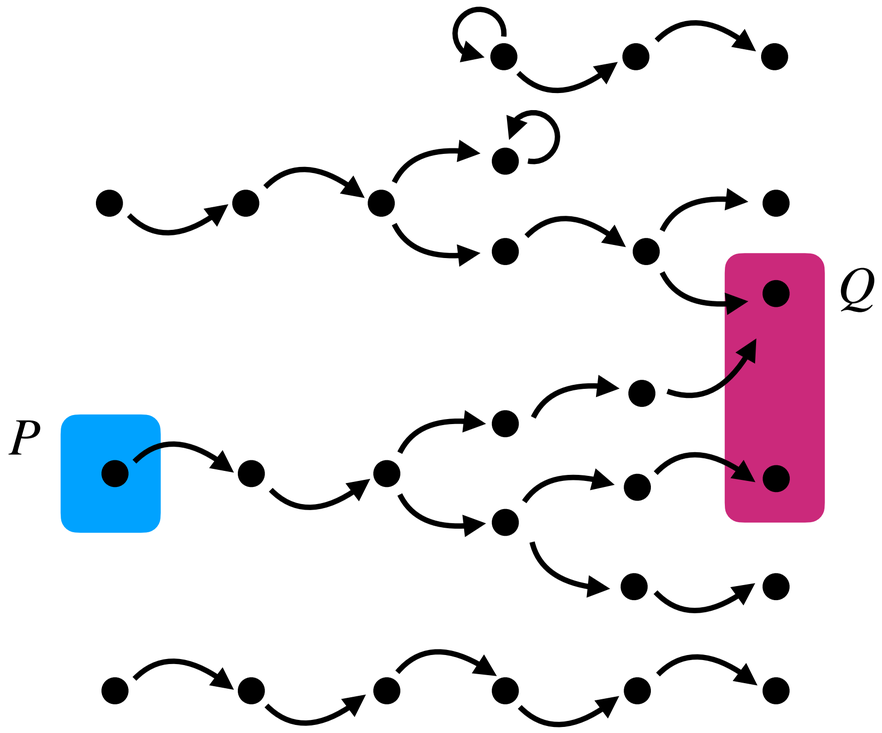


$$\lfloor P \rfloor\ r\ \lfloor Q \rfloor \quad \equiv \quad Q \subseteq [\![\mathsf{r}]\!] P \quad \wedge \quad P \subseteq [\![\overleftarrow{\mathsf{r}}]\!] Q$$

- ➡ Every state in Q is reachable
- ➡ Every state in P is guaranteed to reach a state in Q
- ➡ If Q are errors, P are some source of errors

Under-under

Also studied in [OOPSLA24] UnTer and [POPL26] U-Turn



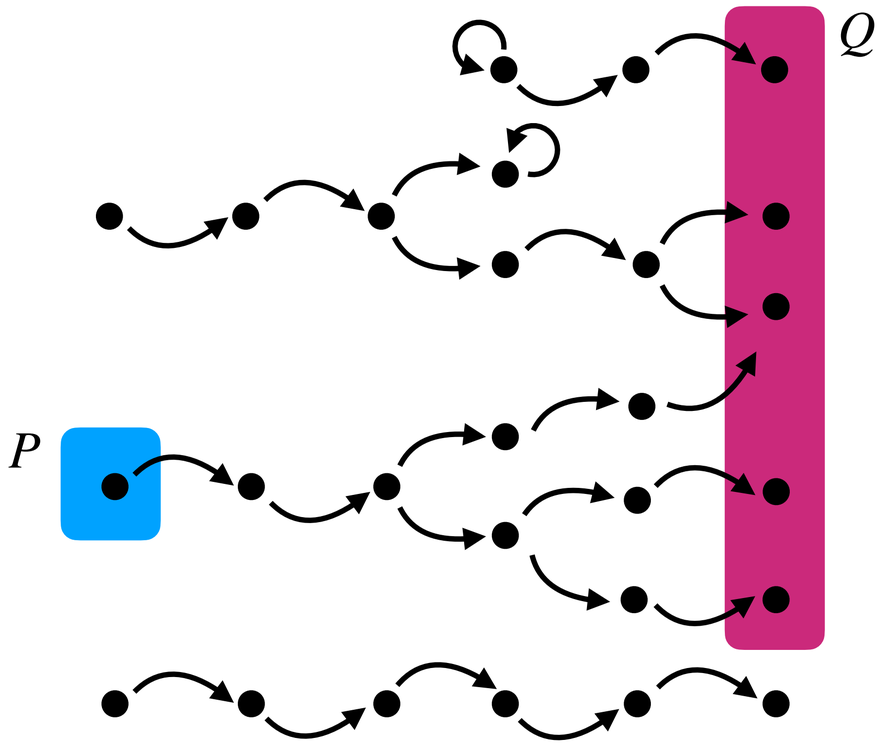
$$\lfloor P \rfloor\ r\ \lfloor Q \rfloor \quad \equiv \quad Q \subseteq [\![\mathsf{r}]\!] P \quad \wedge \quad P \subseteq [\![\overleftarrow{\mathsf{r}}]\!] Q$$

- ➡ Every state in Q is reachable
- ➡ Every state in P is guaranteed to reach a state in Q
- ➡ If Q are errors, P are some source of errors

```
shuffle ≜ a := copy(b);
          m := 0;
          swap(a[m++], a[nondet(0, a.len − 1)])^k
```

$\lfloor \mathtt{k = b.len} \rfloor$ `shuffle` $\lfloor \mathtt{er : a = b^{\dagger} \wedge m = k = b.len} \rfloor$

Under-over

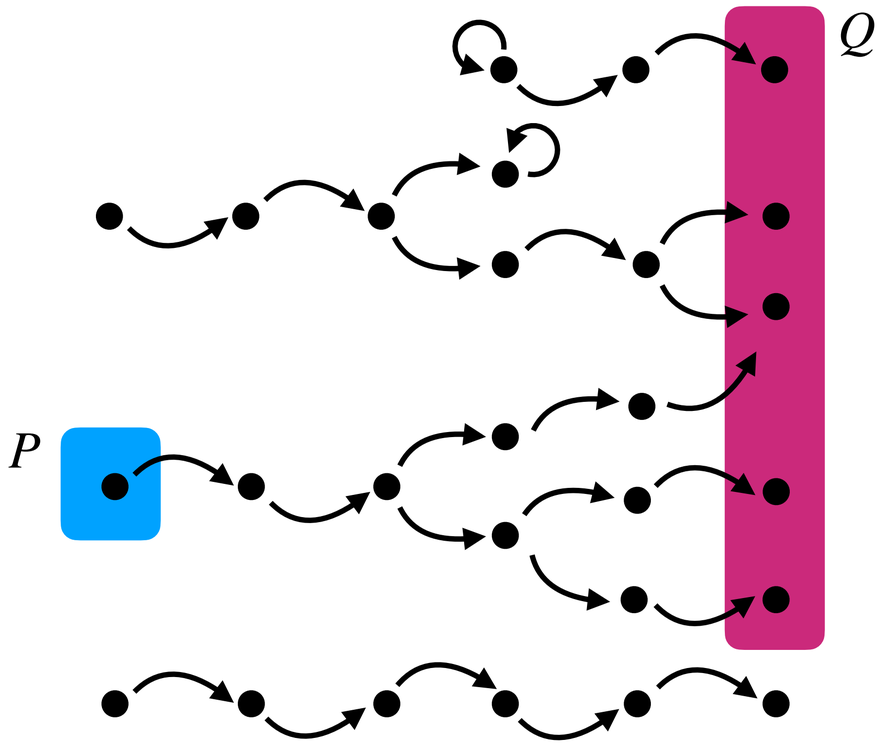


$$\lfloor P \rfloor\ r\ \lceil Q \rceil \quad \equiv \quad [\![\mathsf{r}]\!]P \subseteq Q \quad \wedge \quad P \subseteq [\![\overleftarrow{\mathsf{r}}]\!]Q$$

➡ From P we can only reach Q

➡ Every state in P is guaranteed to reach a state in Q

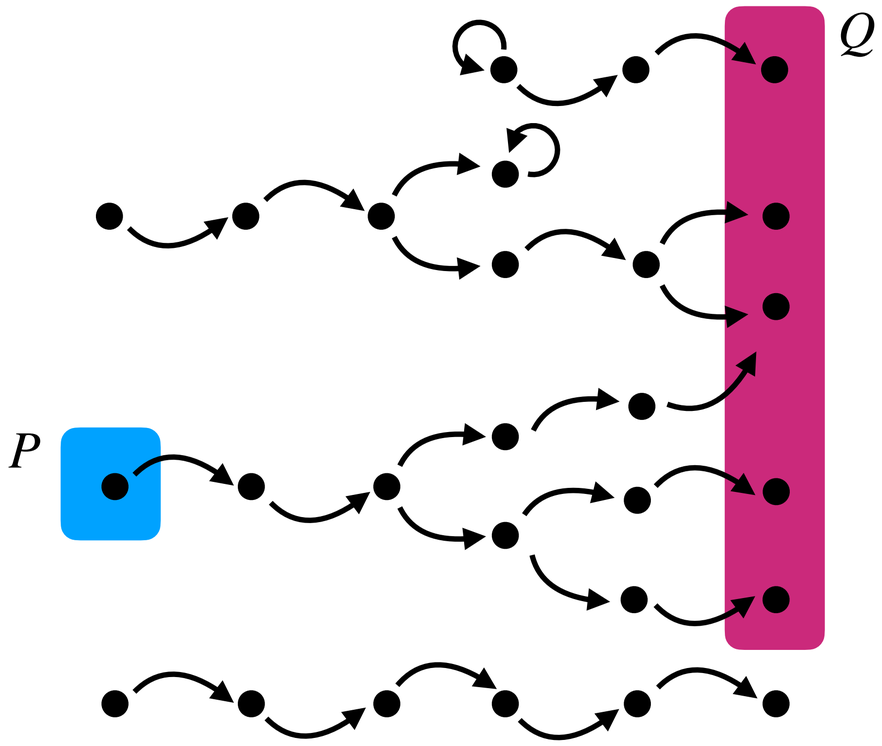
Under-over



$$\lfloor P \rfloor\ r\ \lceil Q \rceil \quad \equiv \quad [\![\mathsf{r}]\!]P \subseteq Q \quad \wedge \quad P \subseteq [\![\overleftarrow{\mathsf{r}}]\!]Q$$

- ➡ From P we can only reach Q
- ➡ Every state in P is guaranteed to reach a state in Q
- ➡ When r is deterministic, it recovers **total correctness**

Under-over



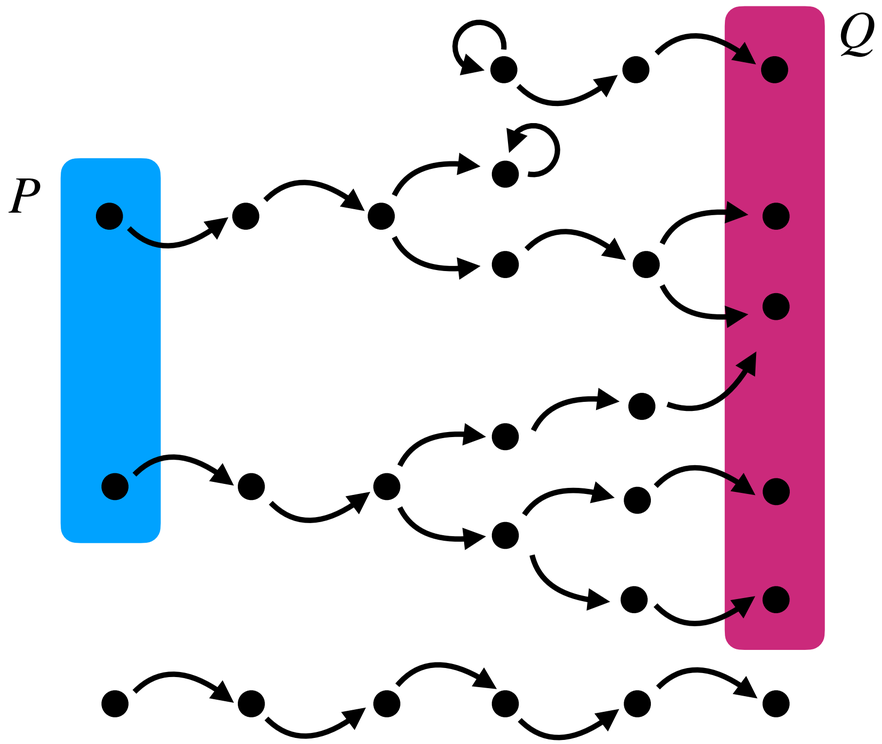
$$\lfloor P \rfloor\ r\ \lceil Q \rceil \quad \equiv \quad [\![r]\!]P \subseteq Q \quad \wedge \quad P \subseteq [\![\overleftarrow{r}]\!]Q$$

- ➡ From P we can only reach Q
- ➡ Every state in P is guaranteed to reach a state in Q
- ➡ When r is deterministic, it recovers **total correctness**

$$\texttt{shuffle} \triangleq \texttt{a} := \texttt{copy(b)};\ \texttt{m} := 0;\ \texttt{swap(a[m++], a[nondet(0, a.len} - 1\texttt{)])}^{\texttt{k}}$$

$$\lfloor \texttt{k} = \texttt{b.len/2} \rfloor\ \texttt{shuffle}\ \lceil \texttt{a} \in \texttt{perm(b)} \rceil$$

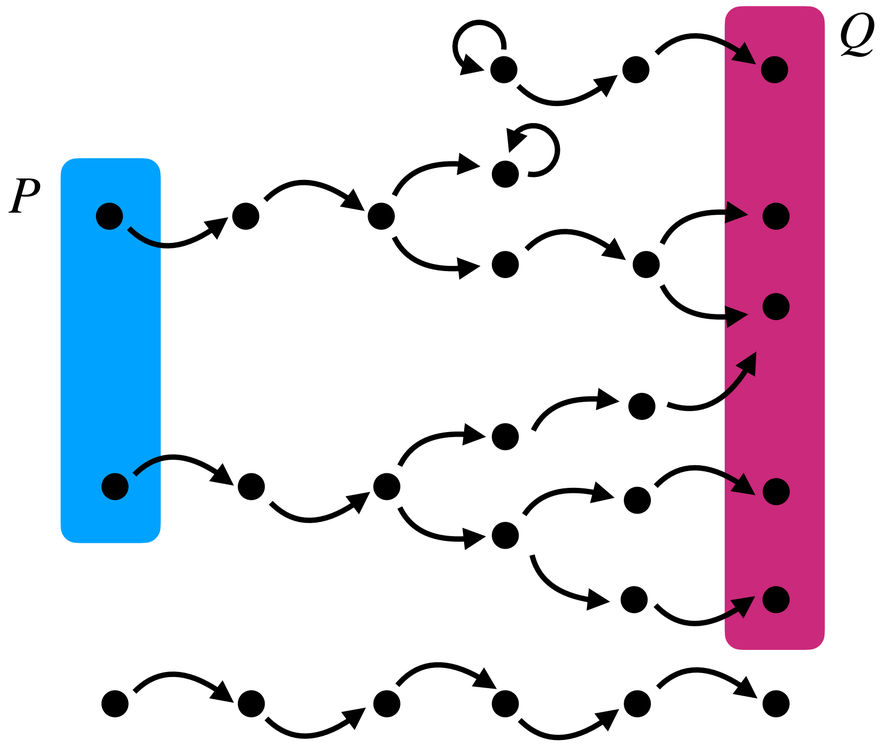
Over-over



$$\lceil P \rceil \; r \; \lceil Q \rceil \quad \equiv \quad [\![r]\!] P \subseteq Q \quad \wedge \quad [\![\overleftarrow{r}]\!] Q \subseteq P$$

- ➡ From P we can only reach Q
- ➡ We can reach Q only if we start from P

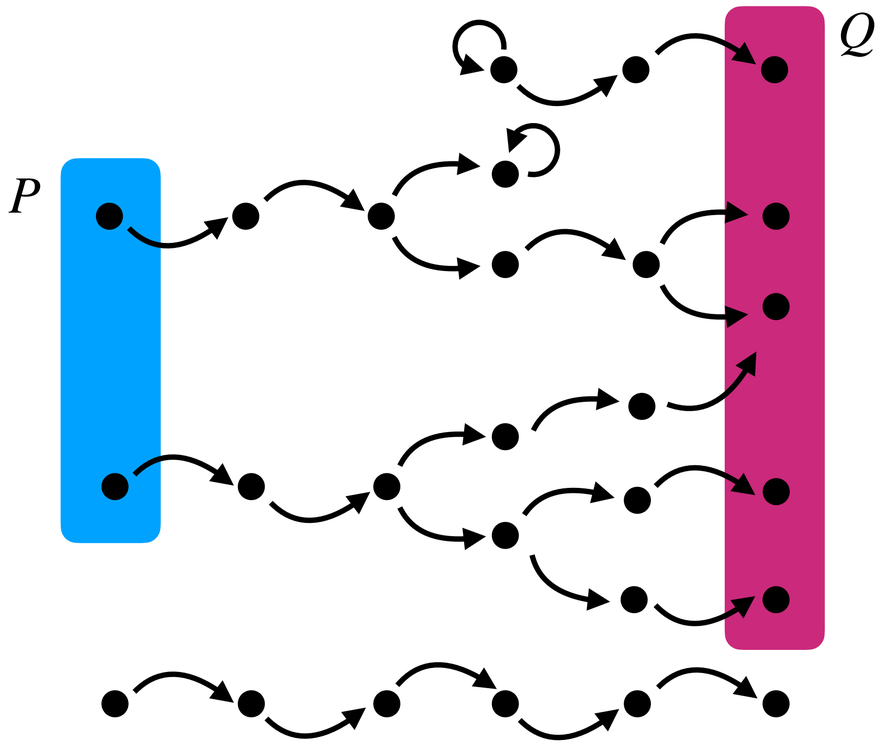
Over-over



$\lceil P \rceil \; r \; \lceil Q \rceil \quad \equiv \quad [\![r]\!] P \subseteq Q \quad \wedge \quad [\![\overleftarrow{r}]\!] Q \subseteq P$

- From P we can only reach Q
- We can reach Q only if we start from P
- From $\neg P$ we can only reach $\neg Q$

Over-over



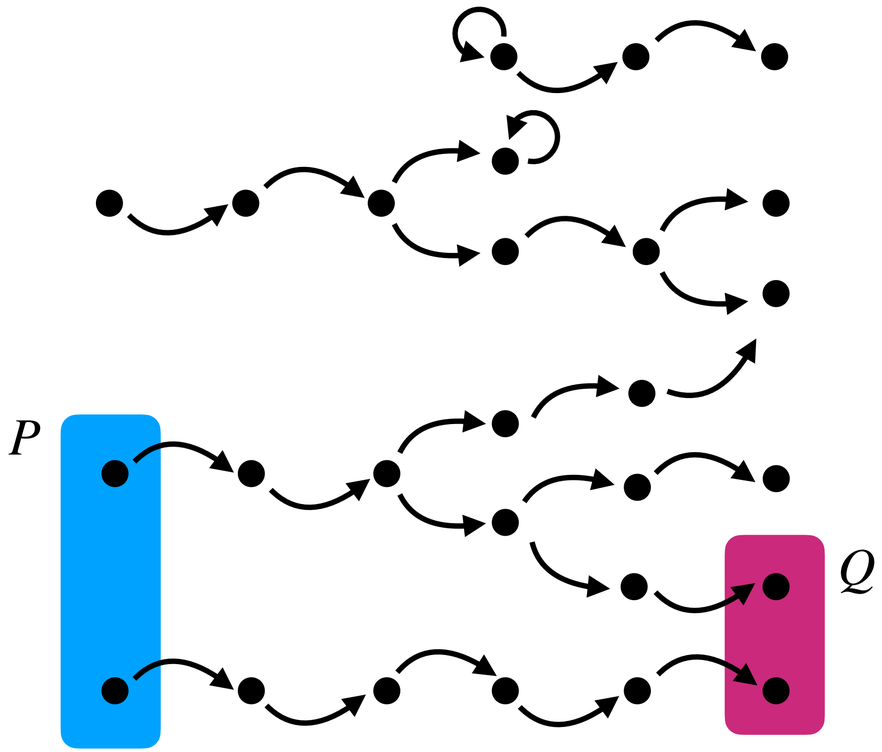
$$\lceil P \rceil\ r\ \lceil Q \rceil \quad \equiv \quad [\![\mathsf{r}]\!]P \subseteq Q \quad \wedge \quad [\![\overleftarrow{\mathsf{r}}]\!]Q \subseteq P$$

- ➡ From P we can only reach Q
- ➡ We can reach Q only if we start from P
- ➡ From $\neg P$ we can only reach $\neg Q$

```
shuffle ≜ a := copy(b);
          m := 0;
          swap(a[m++], a[nondet(0, a.len − 1)])^k
```

$$\lceil \mathtt{k} < \mathtt{b.len} \rceil\ \mathtt{shuffle}\ \lceil \mathtt{a} \in \mathtt{perm(b)} \rceil$$

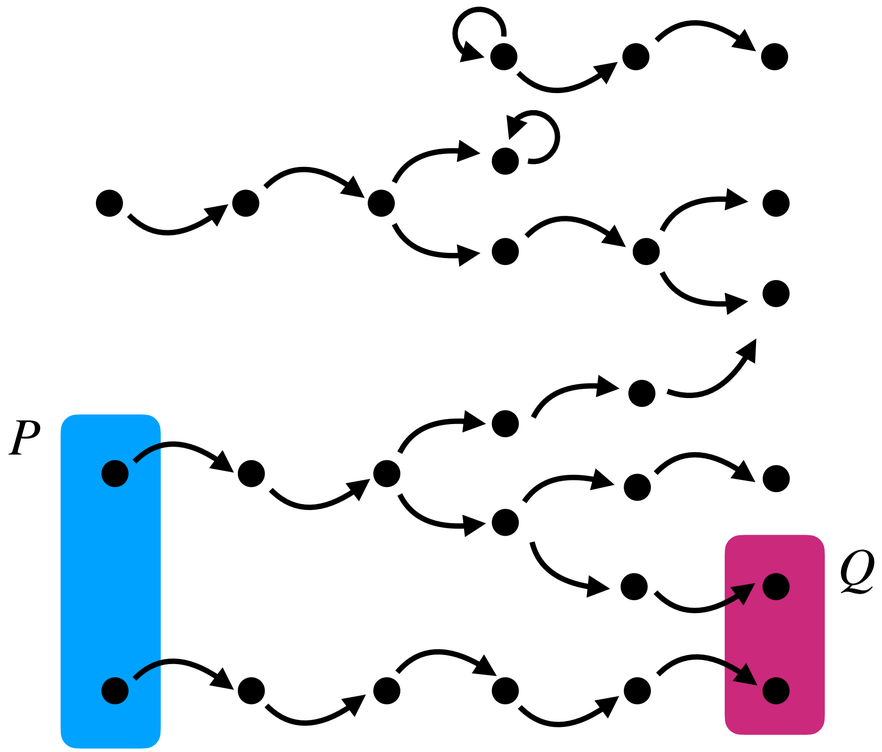
Over-under



$$\lceil P \rceil \; r \; \lfloor Q \rfloor \quad \equiv \quad Q \subseteq [\![\mathsf{r}]\!] P \quad \wedge \quad [\![\overleftarrow{\mathsf{r}}]\!] Q \subseteq P$$

- ➡ Every state in Q is reachable
- ➡ States in Q are reachable only from P

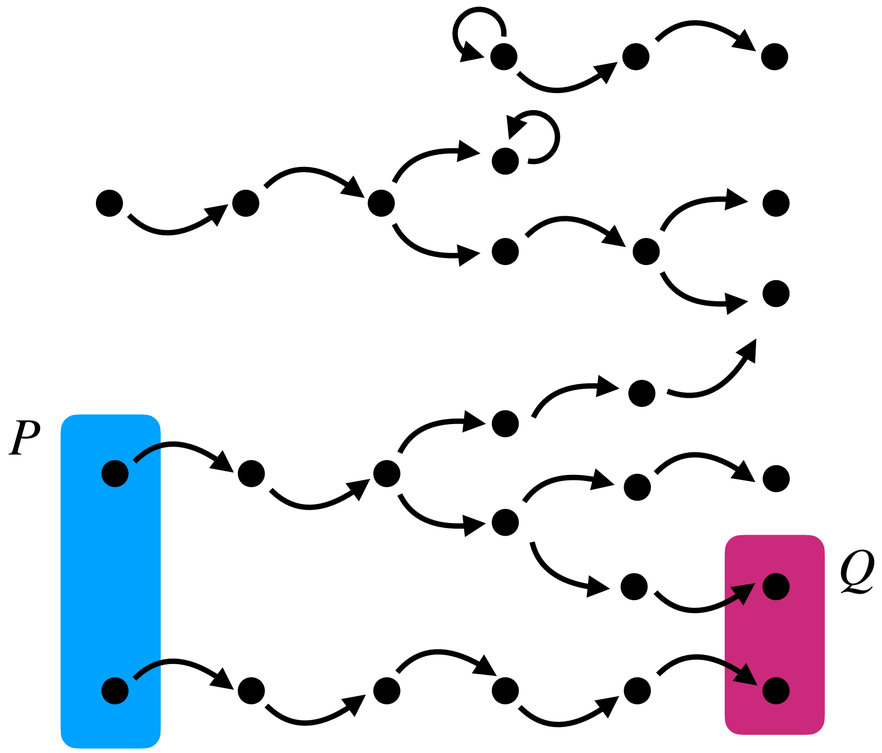
Over-under



$$\lceil P \rceil\ r\ \lfloor Q \rfloor \quad \equiv \quad Q \subseteq [\![\mathsf{r}]\!] P \quad \wedge \quad [\![\overleftarrow{\mathsf{r}}]\!] Q \subseteq P$$

- ➡ Every state in Q is reachable
- ➡ States in Q are reachable only from P
- ➡ If Q error states, then P includes all problematic inputs

Over-under



$$\lceil P \rceil\ r\ \lfloor Q \rfloor \quad \equiv \quad Q \subseteq [\![\mathsf{r}]\!]P \quad \wedge \quad [\![\overleftarrow{\mathsf{r}}]\!]Q \subseteq P$$

- ➡ Every state in Q is reachable
- ➡ States in Q are reachable only from P
- ➡ If Q error states, then P includes all problematic inputs

```
shuffle ≜ a := copy(b);
          m := 0;
          swap(a[m++], a[nondet(0, a.len − 1)])^k
```

$\lceil \mathtt{k} \geq \mathtt{b.len} \rceil$ shuffle $\lfloor \mathtt{er} : \mathtt{a} \in \mathtt{perm(b)} \wedge \mathtt{m} = \mathtt{k} \geq \mathtt{b.len} \rfloor$

Exact-exact

$$\texttt{shuffle} \triangleq \texttt{a} := \texttt{copy(b)};\ \texttt{m} := 0;\ \texttt{swap(a[m++], a[nondet(0, a.len} - 1\texttt{)]})^{\texttt{k}}$$

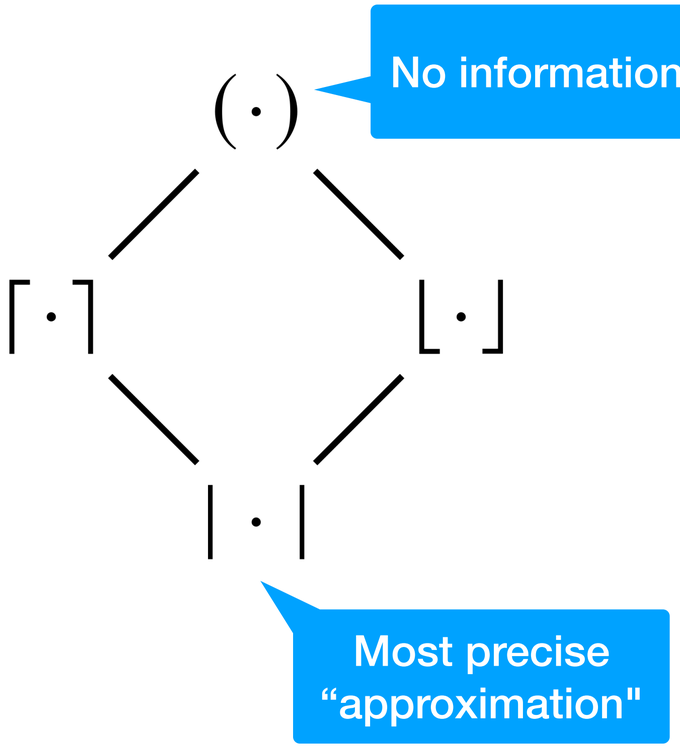
$$|\texttt{k} = \texttt{a.len} - 1|\ \texttt{shuffle}\ |\texttt{a} \in \texttt{perm(b)} \land \texttt{k} = \texttt{m} = \texttt{a.len} - 1|$$

- ➡ From the pre we can reach all the permutations
- ➡ We cannot reach errors
- ➡ The pre is a necessary and sufficient condition to reach only permutations
- ➡ If we start outside the pre we can reach errors

A Logic for All Reasons

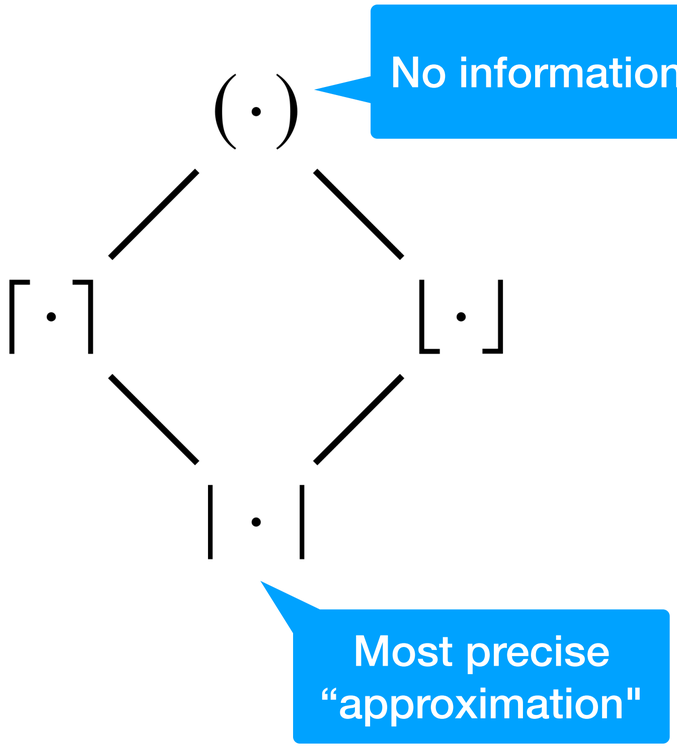
Lattice of approximation

Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$



Lattice of approximation

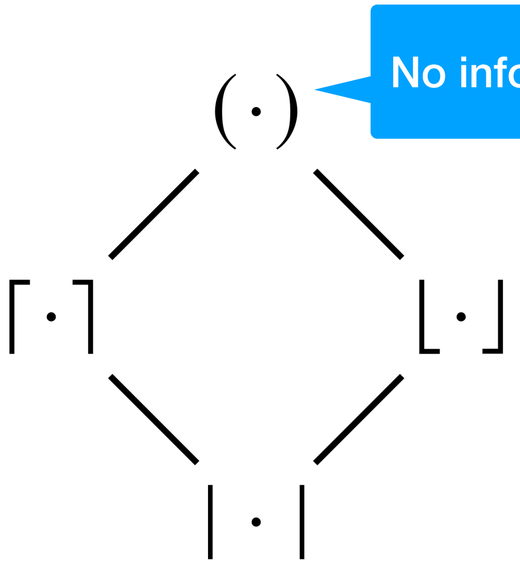
Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$



- $\sqcup$ and $\sqcap$ formalize join and meet of approximations
- We use $\bar{\cdot}$ to reverse parenthesis, e.g., $\overline{\lfloor \cdot \rfloor} = \lceil \cdot \rceil$

Lattice of approximation

Over-approximation	Under-approximation	Exact-approximation	Unknown-approximation
$\lceil \cdot \rceil$	$\lfloor \cdot \rfloor$	$\lvert \cdot \rvert$	$(\cdot)$



No information

Most precise "approximation"

- $\sqcup$ and $\sqcap$ formalize join and meet of approximations
- We use $\bar{\cdot}$ to reverse parenthesis, e.g., $\overline{\lfloor \cdot \rfloor} = \lceil \cdot \rceil$

$P : (\!| R |\!)_\alpha$

P is a α-approximation of R

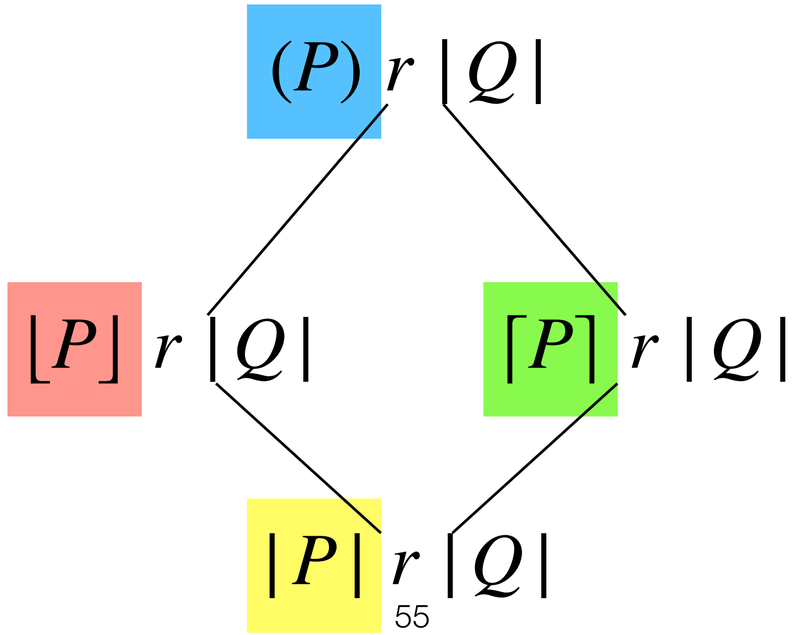
Typing assertions

$$P : \lceil R \rceil \iff R \subseteq P$$
$$P : \lfloor R \rfloor \iff P \subseteq R$$
$$P : \lvert R \rvert \iff R = P$$
$$P : (R) \iff \text{no relation}$$

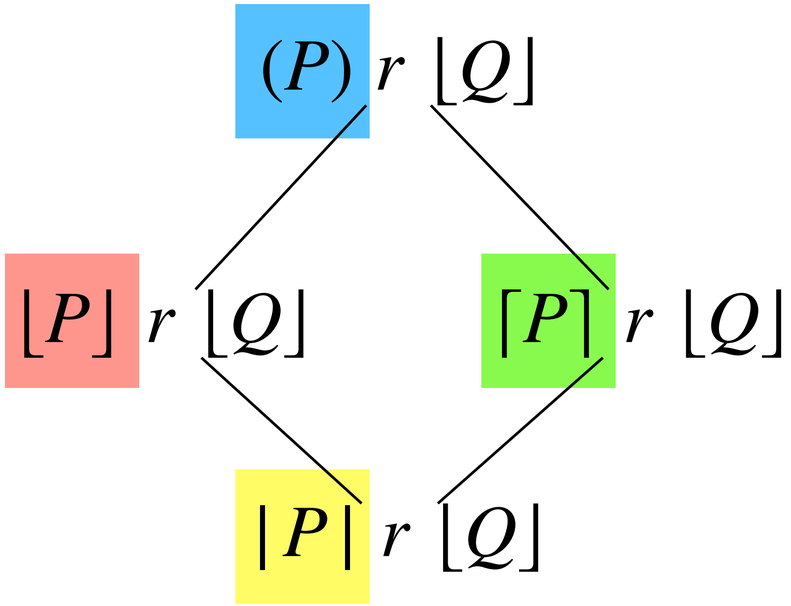
Triple lattice

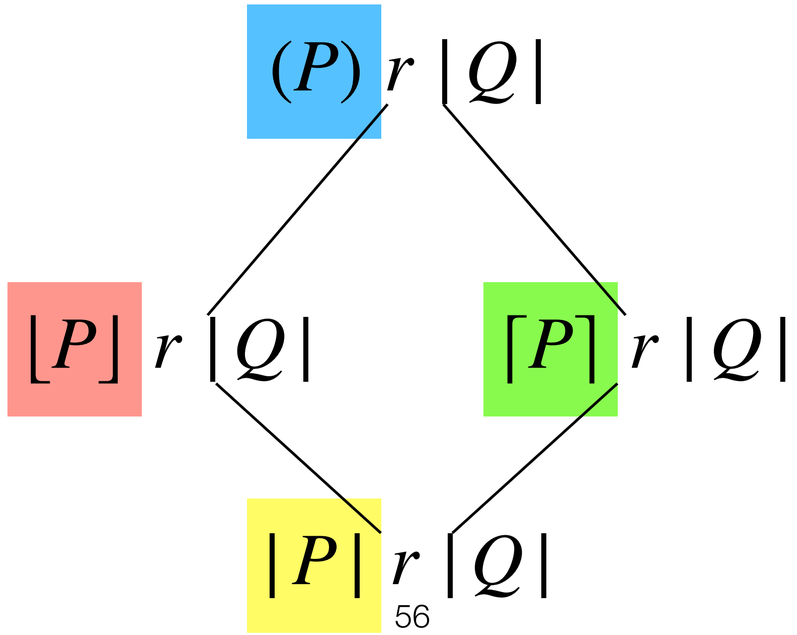
$$|P|\ r\ |Q|$$

Triple lattice

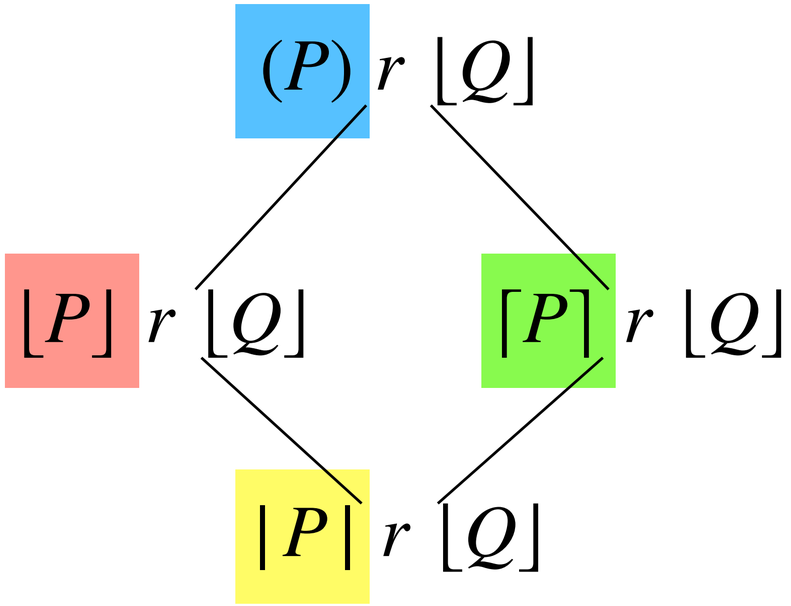


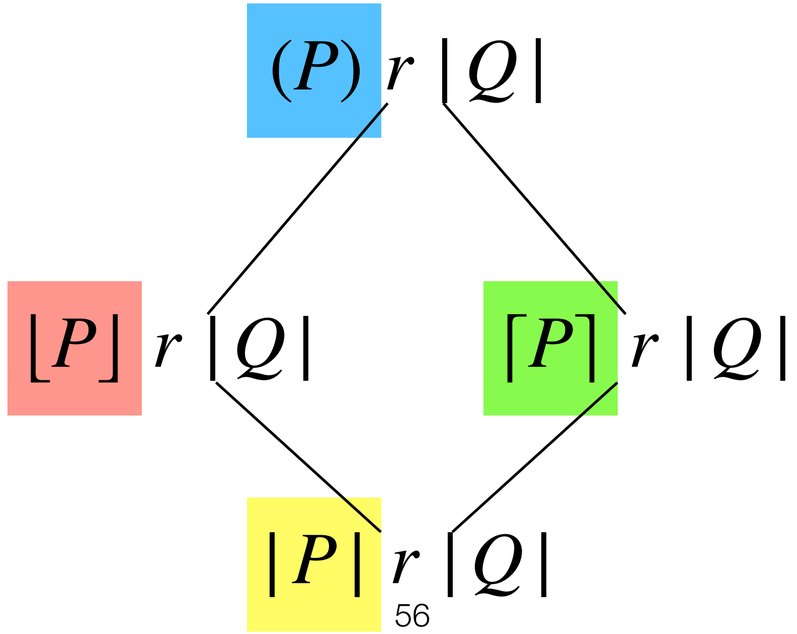
Triple lattice

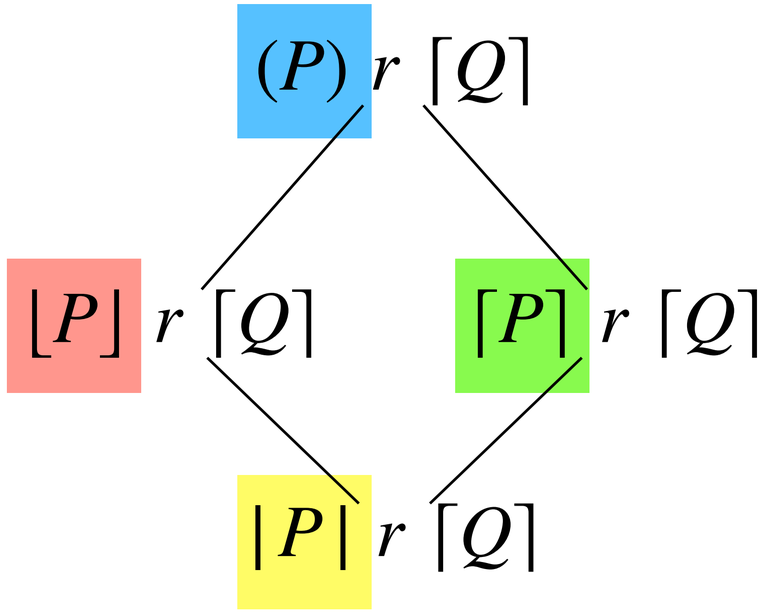




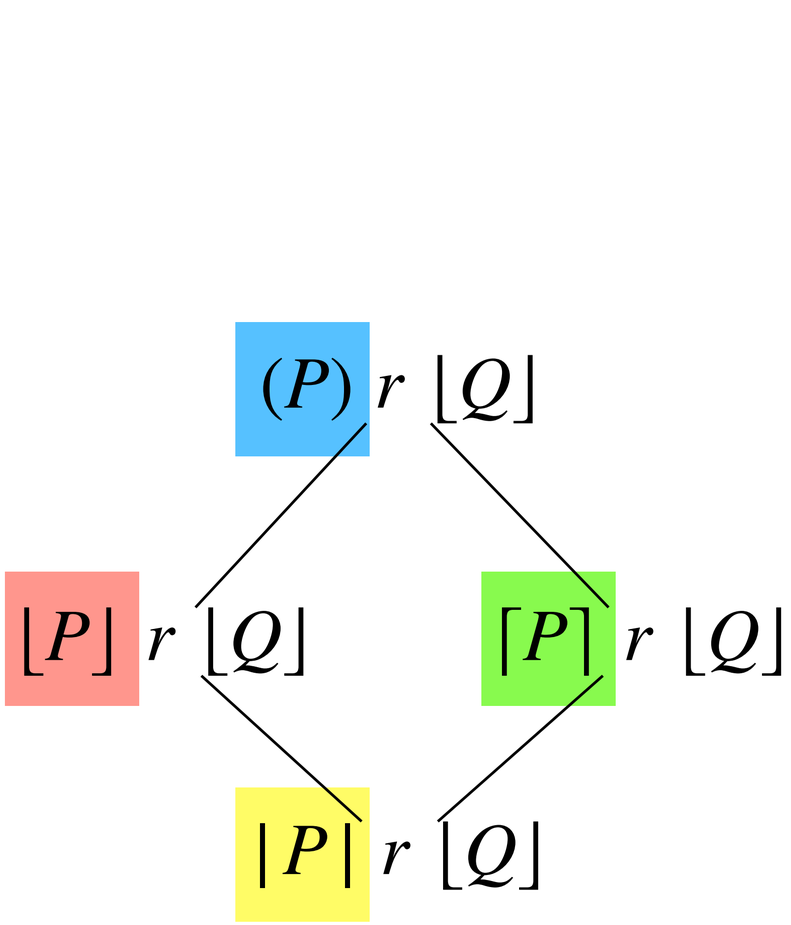
Triple lattice

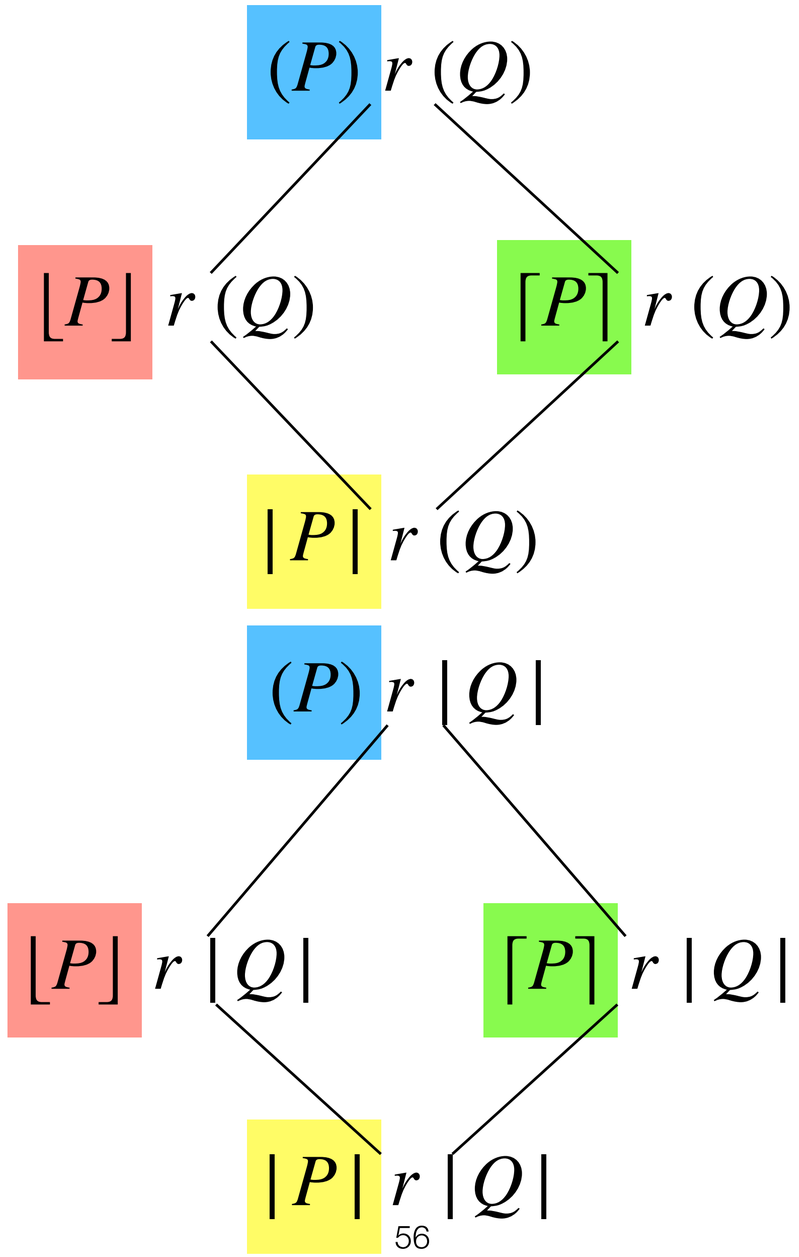


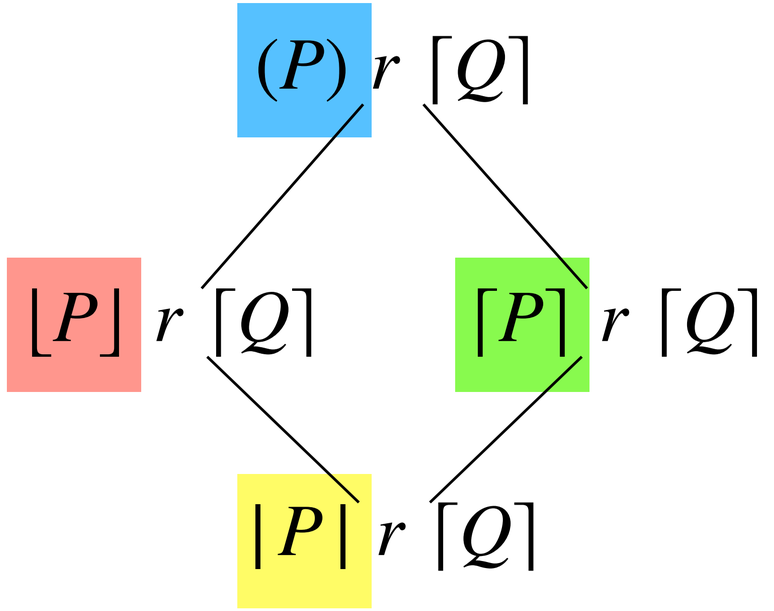




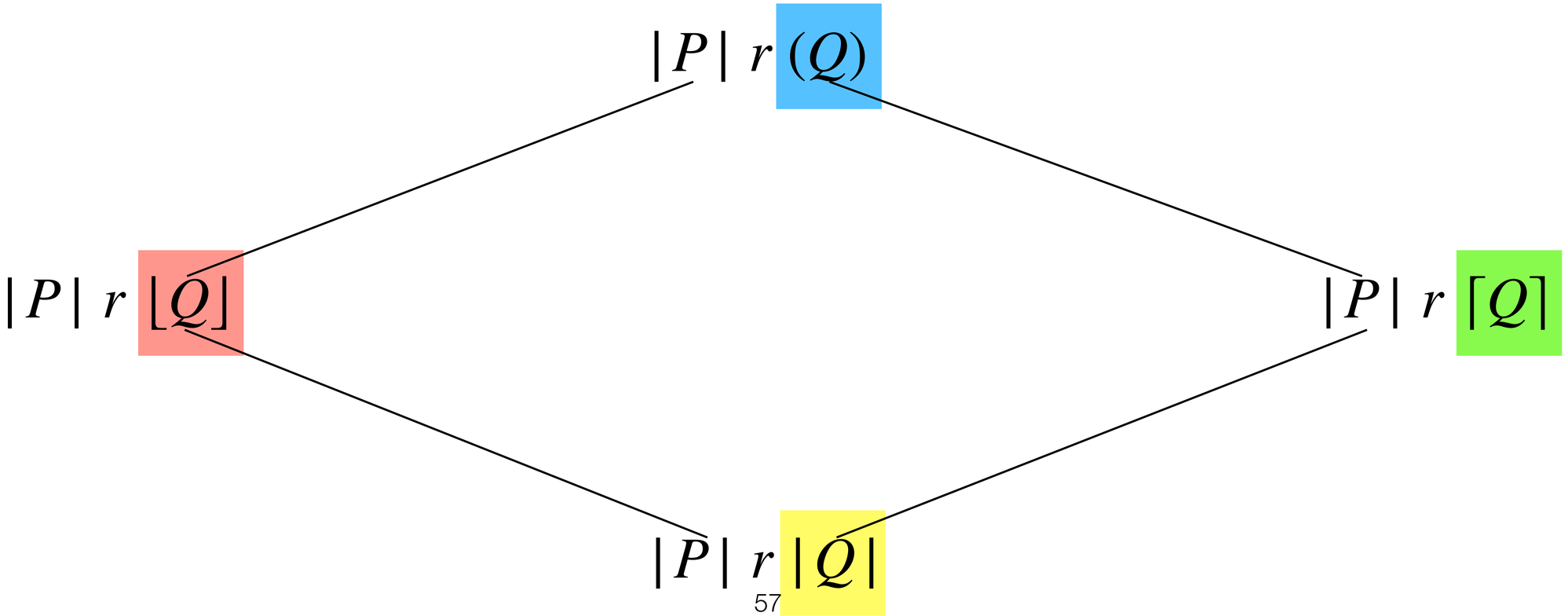
Triple lattice



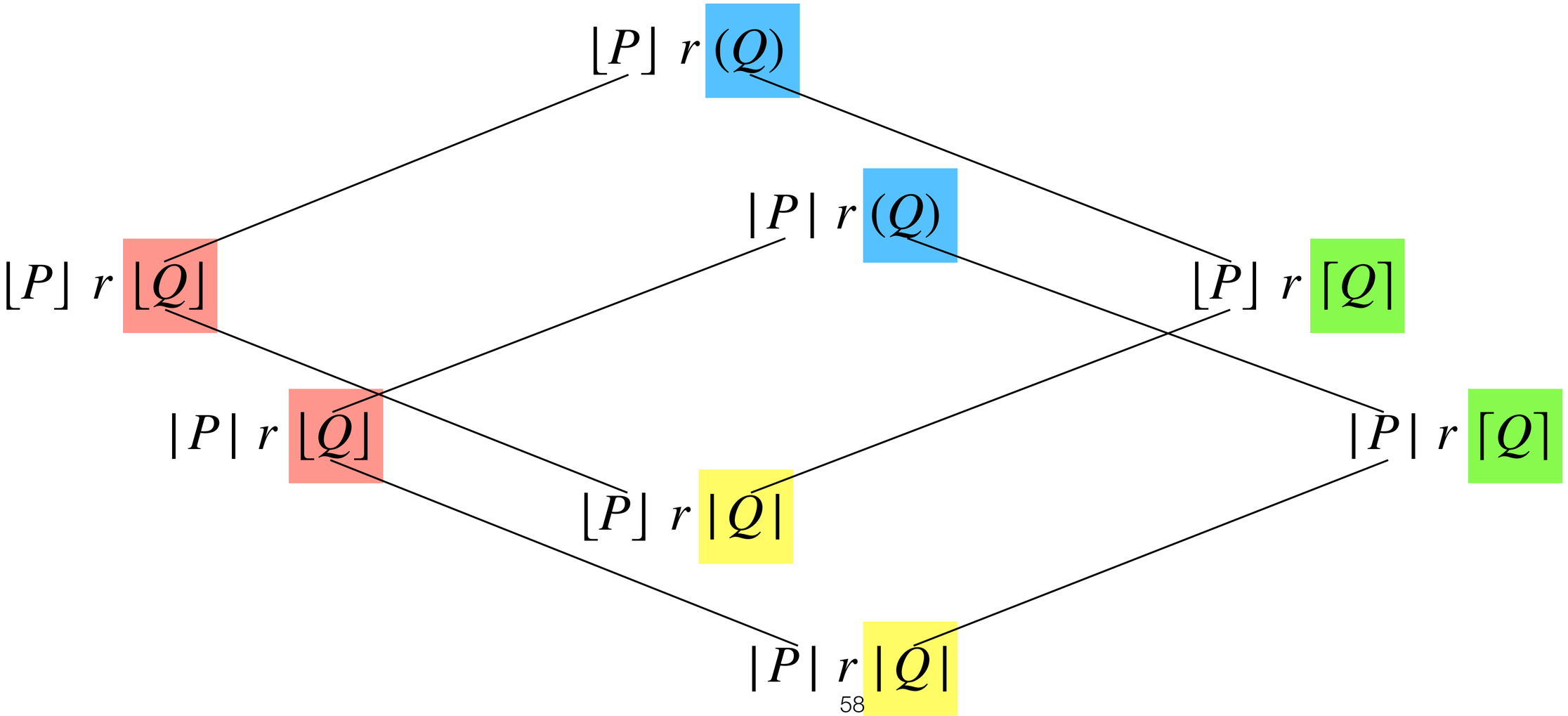




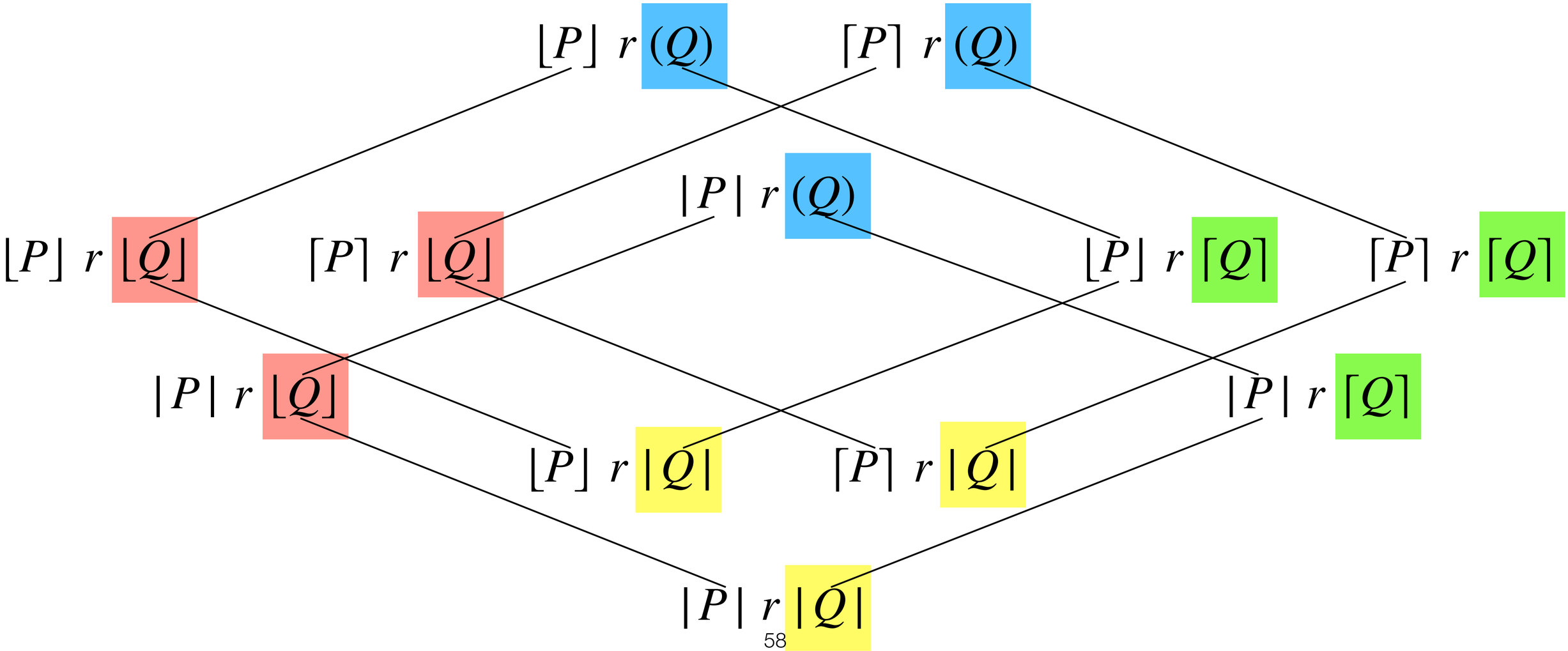
Triple lattice



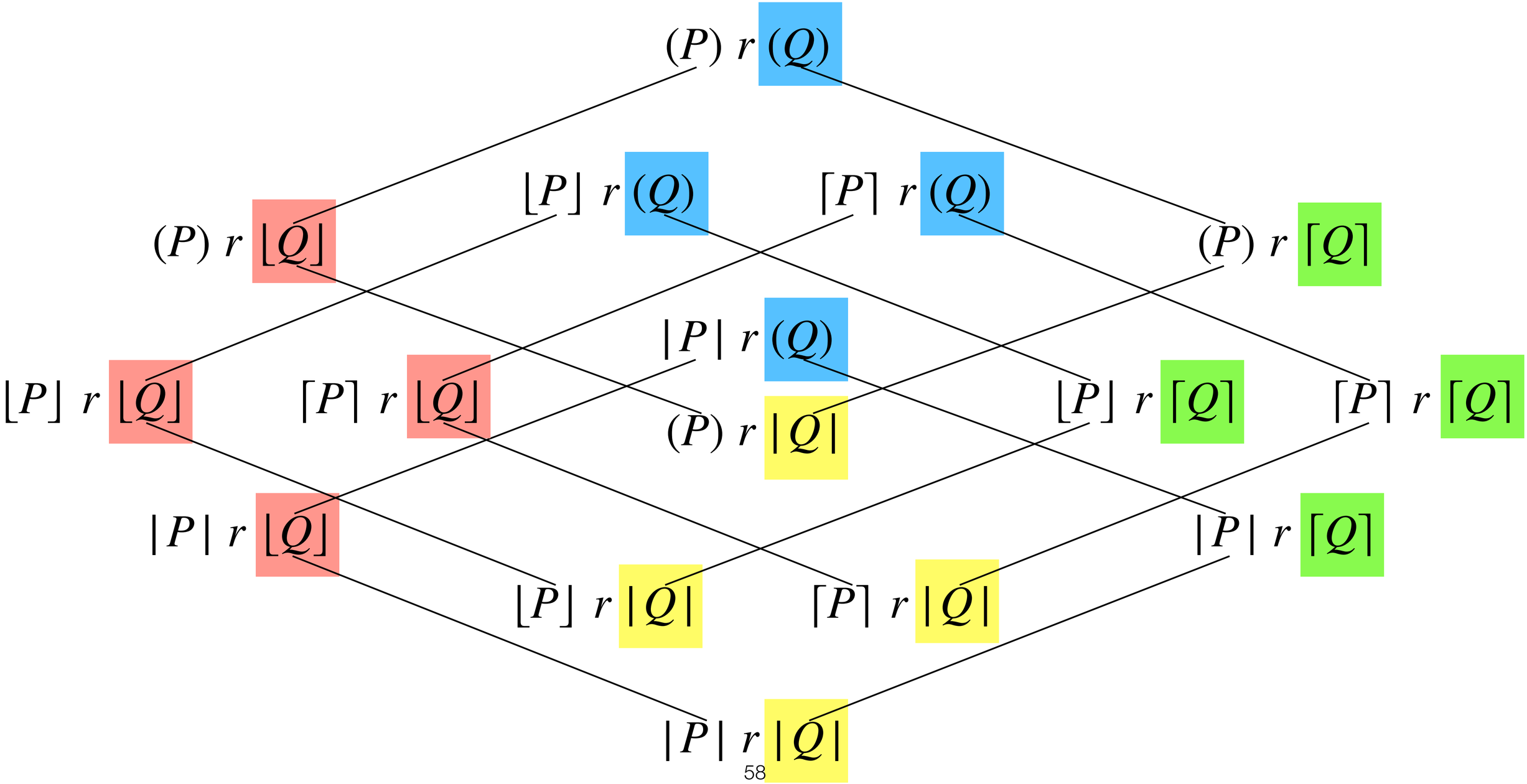
Triple lattice



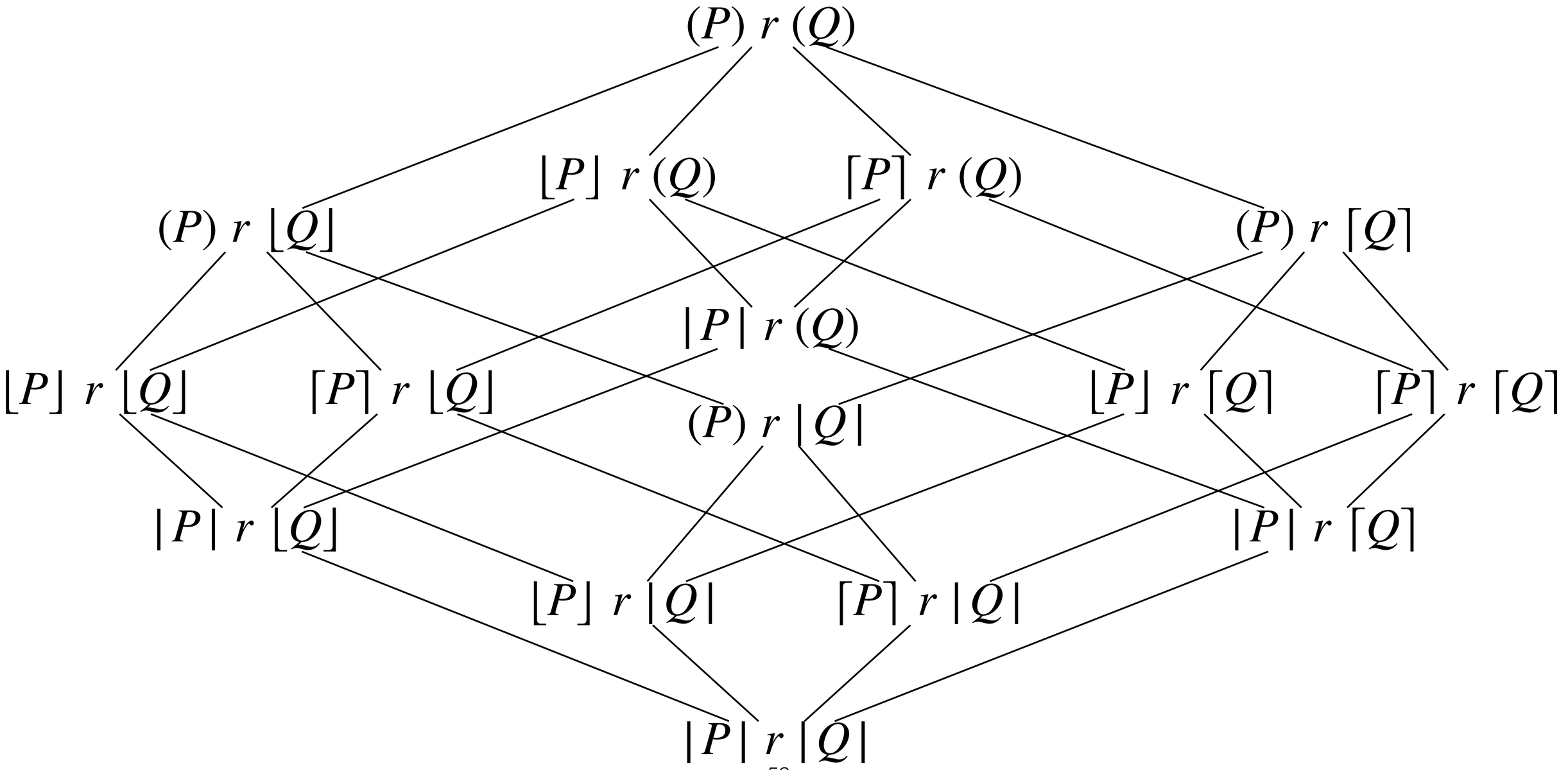
Triple lattice



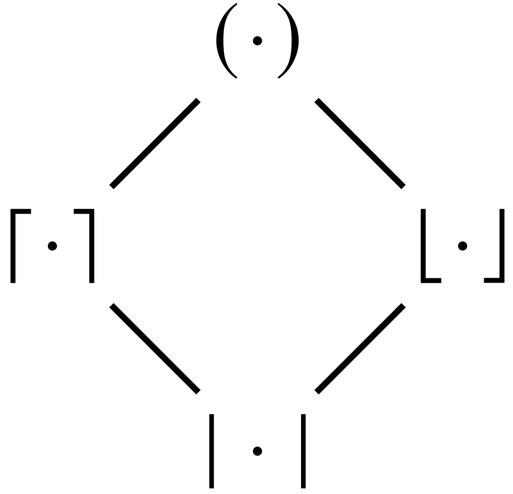
Triple lattice

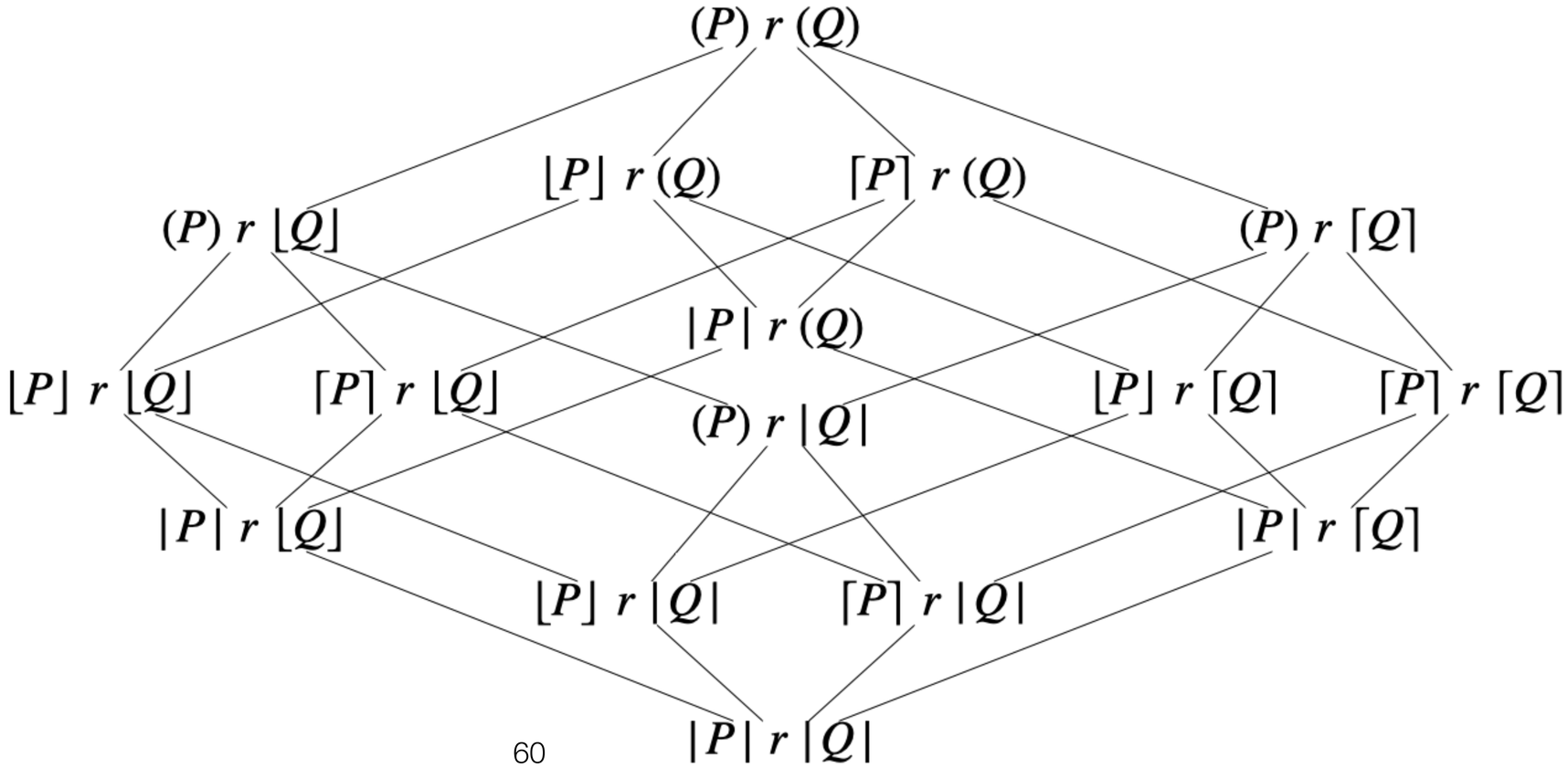


Triple lattice

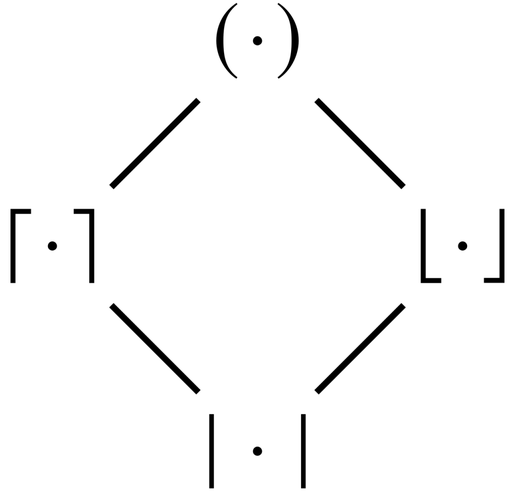


Triple lattice

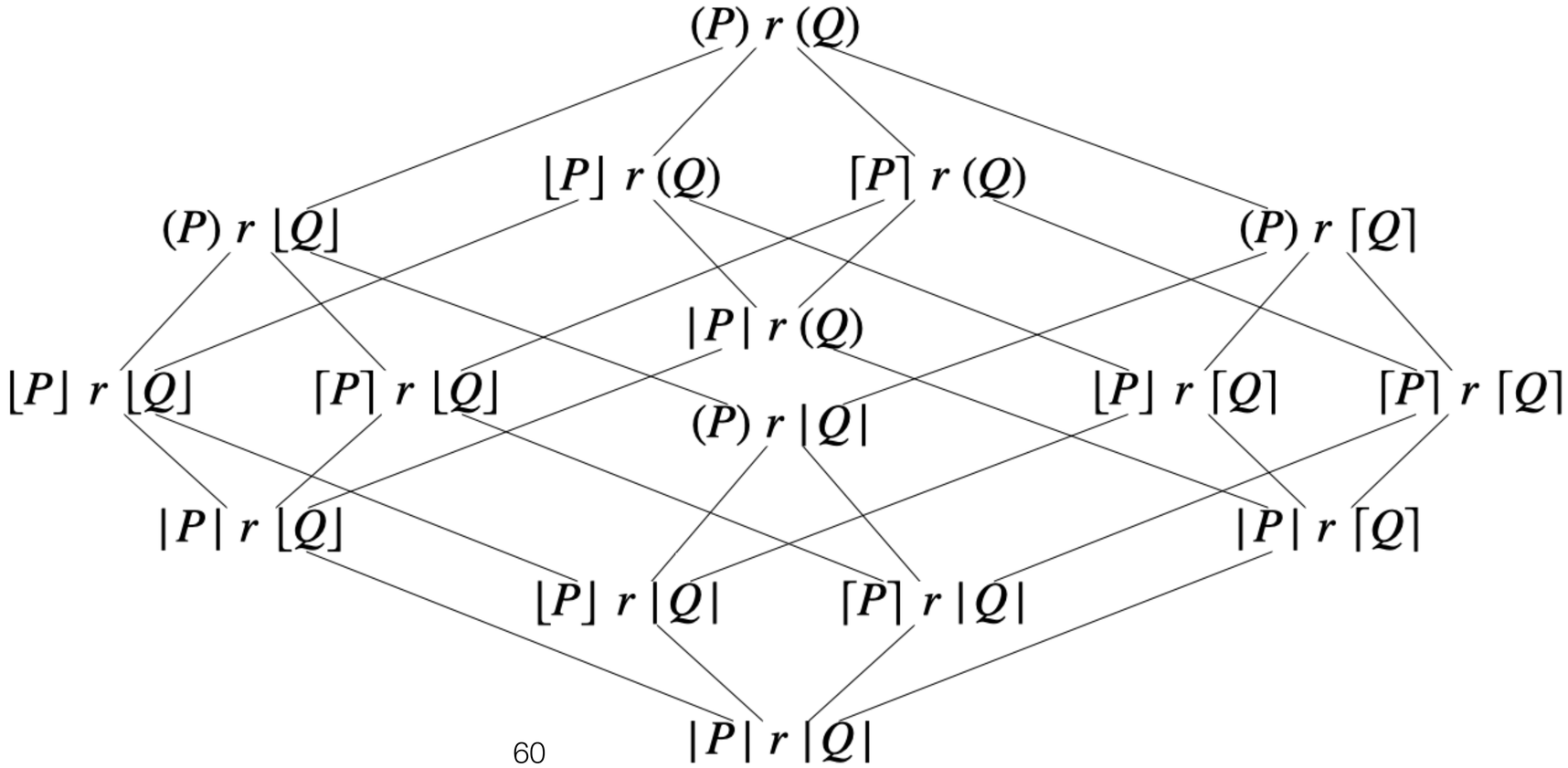




Triple lattice



$$\frac{\alpha \sqsubseteq \gamma \qquad (\!| P |\!)_{\alpha}\ r\ (\!| Q |\!)_{\beta} \qquad \beta \sqsubseteq \delta}{(\!| P |\!)_{\gamma}\ r\ (\!| Q |\!)_{\delta}}\ \text{[Ord]}$$



Sequential composition

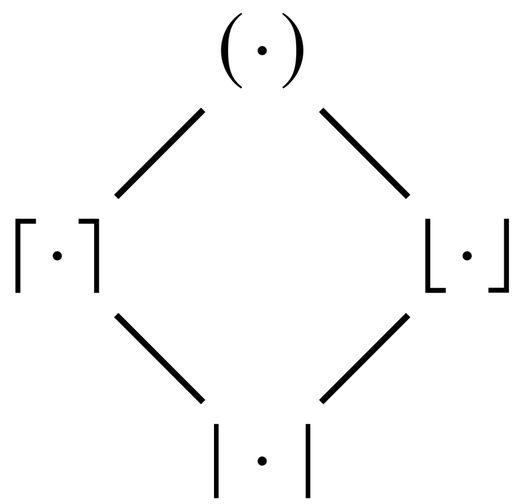
$$\frac{\{P\}\ r_1\ \{R\} \quad \{R\}\ r_2\ \{Q\}}{\{P\}\ r_1; r_2\ \{Q\}}\ \mathsf{HLseq} \qquad \frac{[P]\ r_1\ [R] \quad [R]\ r_2\ [Q]}{[P]\ r_1; r_2\ [Q]}\ \mathsf{ILseq} \qquad \frac{\langle P\rangle\ r_1\ \langle R\rangle \quad \langle R\rangle\ r_2\ \langle Q\rangle}{\langle P\rangle\ r_1; r_2\ \langle Q\rangle}\ \mathsf{SILseq}$$

Sequential composition

$$\frac{(P)\ r_1\ \lceil R \rceil \quad (R)\ r_2\ \lceil Q \rceil}{(P)\ r_1; r_2\ \lceil Q \rceil}\ \text{HLseq}$$

$$\frac{(P)\ r_1\ \lfloor R \rfloor \quad (R)\ r_2\ \lfloor Q \rfloor}{(P)\ r_1; r_2\ \lfloor Q \rfloor}\ \text{ILseq}$$

$$\frac{\lfloor P \rfloor\ r_1\ (R) \quad \lfloor R \rfloor\ r_2\ (Q)}{\lfloor P \rfloor\ r_1; r_2\ (Q)}\ \text{SILseq}$$



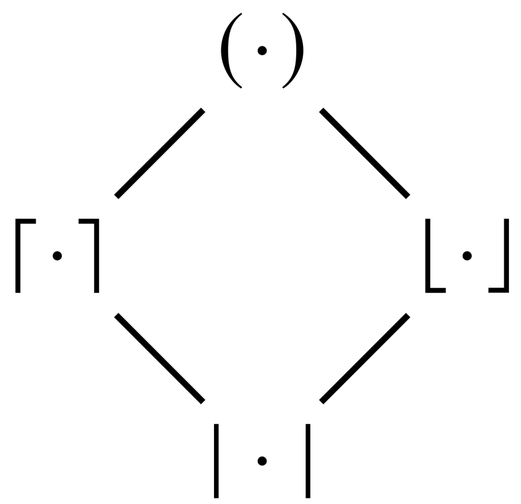
Using standard braces

Sequential composition

$$\frac{(P)\ r_1\ \lceil R \rceil \quad (R)\ r_2\ \lceil Q \rceil}{(P)\ r_1; r_2\ \lceil Q \rceil}\ \text{HLseq}$$

$$\frac{(P)\ r_1\ \lfloor R \rfloor \quad (R)\ r_2\ \lfloor Q \rfloor}{(P)\ r_1; r_2\ \lfloor Q \rfloor}\ \text{ILseq}$$

$$\frac{\lfloor P \rfloor\ r_1\ (R) \quad \lfloor R \rfloor\ r_2\ (Q)}{\lfloor P \rfloor\ r_1; r_2\ (Q)}\ \text{SILseq}$$



Using standard braces

$$\frac{(\!|\ P\ |\!)_{\alpha}\ r_1\ (\!|\ R\ |\!)_{\beta} \quad (\!|\ R\ |\!)_{\gamma}\ r_2\ (\!|\ Q\ |\!)_{\delta}}{(\!|\ P\ |\!)_{\alpha \sqcup \gamma}\ r_1; r_2\ (\!|\ Q\ |\!)_{\beta \sqcup \delta}}\ [\text{Seq}]$$

A single rule (for all triples!): 256 cases!

Consequence rule

$$\frac{P \Rightarrow P' \quad \{P'\}\ r\ \{Q'\} \quad Q' \Rightarrow Q}{\{P\}\ r\ \{Q\}}\ \text{HLseq}$$

$$\frac{P \Rightarrow P' \quad \langle P'\rangle\ r\ \langle Q'\rangle \quad Q' \Rightarrow Q}{\langle P\rangle\ r\ \langle Q\rangle}\ \text{SILseq}$$

$$\frac{P' \Rightarrow P \quad [P']\ r\ [Q'] \quad Q \Rightarrow Q'}{[P]\ r\ [Q]}\ \text{ILseq}$$

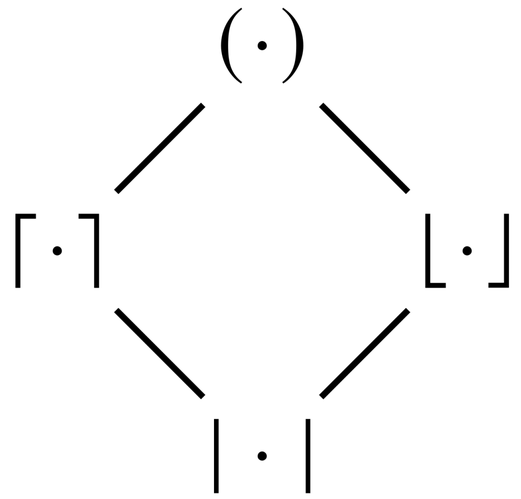
Consequence rule

$$\frac{P : \lfloor P' \rceil \quad (P')\ r\ \lceil Q' \rceil \quad Q : \lceil Q' \rceil}{(P)\ r\ \lceil Q \rceil}\ \text{HLseq}$$

$$\frac{P : \lfloor P' \rceil \quad \lfloor P' \rceil\ r\ (Q') \quad Q : \lceil Q \rceil}{\lfloor P \rceil\ r\ (Q)}\ \text{SILseq}$$

$$\frac{P : \lceil P' \rceil \quad (P')\ r\ \lfloor Q' \rceil \quad Q : \lfloor Q \rfloor}{(P)\ r\ \lfloor Q \rfloor}\ \text{ILseq}$$

Using standard braces



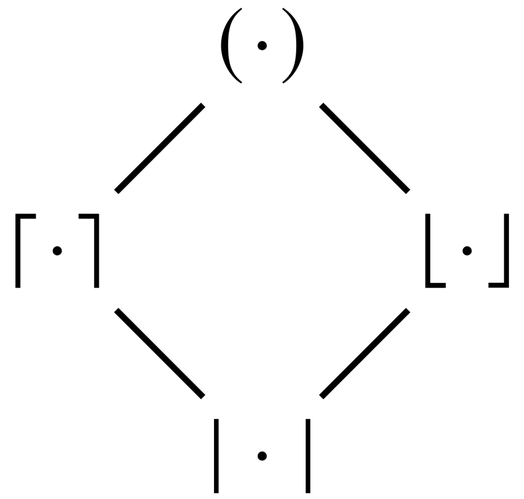
Consequence rule

$$\frac{P : \lfloor P' \rfloor \quad (P')\ r\ \lceil Q' \rceil \quad Q : \lceil Q' \rceil}{(P)\ r\ \lceil Q \rceil}\ \text{HLseq}$$

$$\frac{P : \lfloor P' \rfloor \quad \lfloor P' \rfloor\ r\ (Q') \quad Q : \lceil Q \rceil}{\lfloor P \rfloor\ r\ (Q)}\ \text{SILseq}$$

$$\frac{P : \lceil P' \rceil \quad (P')\ r\ \lfloor Q' \rfloor \quad Q : \lfloor Q \rfloor}{(P)\ r\ \lfloor Q \rfloor}\ \text{ILseq}$$

Using standard braces



$$\frac{P : (\!|\, P' \,|\!)_{\gamma} \qquad (\!|\, P' \,|\!)_{\alpha}\ r\ (\!|\, Q' \,|\!)_{\beta} \qquad Q : (\!|\, Q' \,|\!)_{\delta}}{(\!|\, P \,|\!)_{\alpha \sqcup \gamma \sqcup \bar{\delta}}\ r\ (\!|\, Q \,|\!)_{\beta \sqcup \bar{\gamma} \sqcup \delta}}\ [\text{Cons}]$$

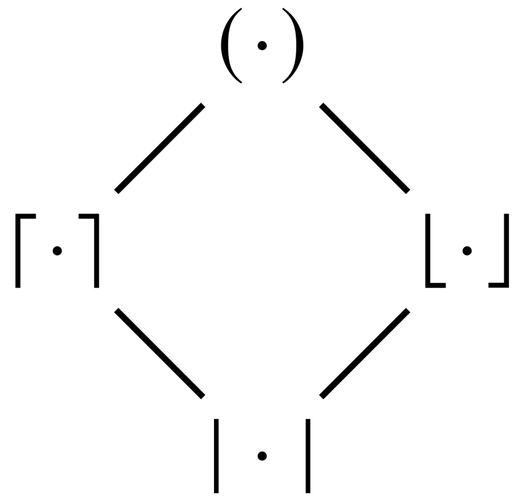
Consequence rule

$$\frac{P : \lfloor P' \rfloor \quad (P')\ r\ \lceil Q' \rceil \quad Q : \lceil Q' \rceil}{(P)\ r\ \lceil Q \rceil}\ \text{HLseq}$$

$$\frac{P : \lfloor P' \rfloor \quad \lfloor P' \rfloor\ r\ (Q') \quad Q : \lceil Q \rceil}{\lfloor P \rfloor\ r\ (Q)}\ \text{SILseq}$$

$$\frac{P : \lceil P' \rceil \quad (P')\ r\ \lfloor Q' \rfloor \quad Q : \lfloor Q \rfloor}{(P)\ r\ \lfloor Q \rfloor}\ \text{ILseq}$$

Using standard braces



$$\frac{P : (\!|\, P' \,|\!)_{\gamma} \qquad (\!|\, P' \,|\!)_{\alpha}\ r\ (\!|\, Q' \,|\!)_{\beta} \qquad Q : (\!|\, Q' \,|\!)_{\delta}}{(\!|\, P \,|\!)_{\alpha \sqcup \gamma \sqcup \bar{\delta}}\ r\ (\!|\, Q \,|\!)_{\beta \sqcup \bar{\gamma} \sqcup \delta}}\ [\text{Cons}]$$

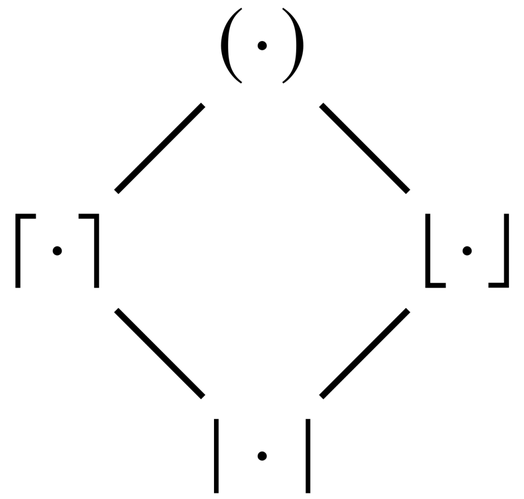
Consequence rule

$$\frac{P : \lfloor P' \rfloor \quad (P')\ r\ \lceil Q' \rceil \quad Q : \lceil Q' \rceil}{(P)\ r\ \lceil Q \rceil}\ \text{HLseq}$$

$$\frac{P : \lfloor P' \rfloor \quad \lfloor P' \rfloor\ r\ (Q') \quad Q : \lceil Q \rceil}{\lfloor P \rfloor\ r\ (Q)}\ \text{SILseq}$$

$$\frac{P : \lceil P' \rceil \quad (P')\ r\ \lfloor Q' \rfloor \quad Q : \lfloor Q \rfloor}{(P)\ r\ \lfloor Q \rfloor}\ \text{ILseq}$$

Using standard braces



$$\frac{P : (\!|\, P' \,|\!)_{\gamma} \qquad (\!|\, P' \,|\!)_{\alpha}\ r\ (\!|\, Q' \,|\!)_{\beta} \qquad Q : (\!|\, Q' \,|\!)_{\delta}}{(\!|\, P \,|\!)_{\alpha \sqcup \gamma \sqcup \bar{\delta}}\ r\ (\!|\, Q \,|\!)_{\beta \sqcup \bar{\gamma} \sqcup \delta}}\ \text{[Cons]}$$

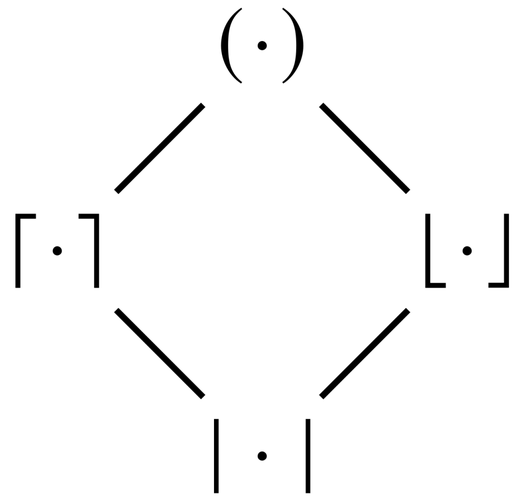
Consequence rule

$$\frac{P : \lfloor P' \rfloor \quad (P')\ r\ \lceil Q' \rceil \quad Q : \lceil Q' \rceil}{(P)\ r\ \lceil Q \rceil}\ \text{HLseq}$$

$$\frac{P : \lfloor P' \rfloor \quad \lfloor P' \rfloor\ r\ (Q') \quad Q : \lceil Q \rceil}{\lfloor P \rfloor\ r\ (Q)}\ \text{SILseq}$$

$$\frac{P : \lceil P' \rceil \quad (P')\ r\ \lfloor Q' \rfloor \quad Q : \lfloor Q \rfloor}{(P)\ r\ \lfloor Q \rfloor}\ \text{ILseq}$$

Using standard braces



$$\frac{P : (\!|\, P' \,|\!)_{\gamma} \qquad (\!|\, P' \,|\!)_{\alpha}\ r\ (\!|\, Q' \,|\!)_{\beta} \qquad Q : (\!|\, Q' \,|\!)_{\delta}}{(\!|\, P \,|\!)_{\alpha \sqcup \gamma \sqcup \bar{\delta}}\ r\ (\!|\, Q \,|\!)_{\beta \sqcup \bar{\gamma} \sqcup \delta}}\ \text{[Cons]}$$

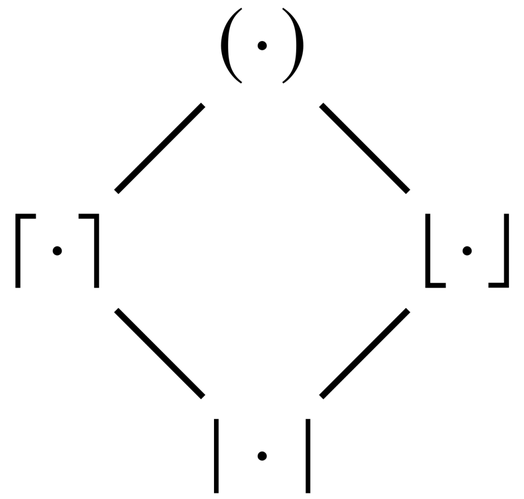
Consequence rule

$$\frac{P : \lfloor P' \rfloor \quad (P')\ r\ \lceil Q' \rceil \quad Q : \lceil Q' \rceil}{(P)\ r\ \lceil Q \rceil}\ \text{HLseq}$$

$$\frac{P : \lfloor P' \rfloor \quad \lfloor P' \rfloor\ r\ (Q') \quad Q : \lceil Q \rceil}{\lfloor P \rfloor\ r\ (Q)}\ \text{SILseq}$$

$$\frac{P : \lceil P' \rceil \quad (P')\ r\ \lfloor Q' \rfloor \quad Q : \lfloor Q \rfloor}{(P)\ r\ \lfloor Q \rfloor}\ \text{ILseq}$$

Using standard braces



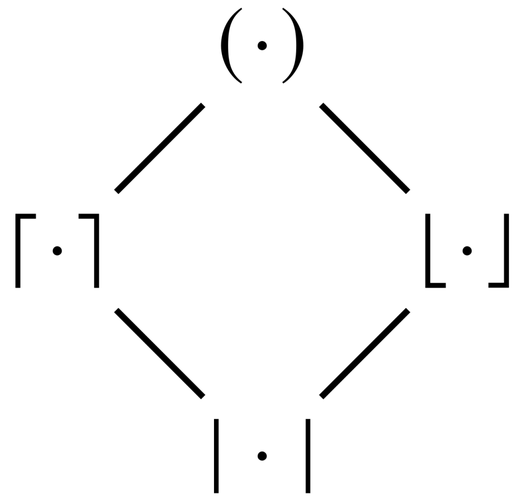
$$\frac{P : (\!|\, P' \,|\!)_{\gamma} \qquad (\!|\, P' \,|\!)_{\alpha}\ r\ (\!|\, Q' \,|\!)_{\beta} \qquad Q : (\!|\, Q' \,|\!)_{\delta}}{(\!|\, P \,|\!)_{\alpha \sqcup \gamma \sqcup \bar{\delta}}\ r\ (\!|\, Q \,|\!)_{\beta \sqcup \bar{\gamma} \sqcup \delta}}\ \text{[Cons]}$$

Consequence rule

$$\frac{P : \lfloor P' \rfloor \quad (P')\ r\ \lceil Q' \rceil \quad Q : \lceil Q' \rceil}{(P)\ r\ \lceil Q \rceil}\ \text{HLseq}$$

$$\frac{P : \lfloor P' \rfloor \quad \lfloor P' \rfloor\ r\ (Q') \quad Q : \lceil Q \rceil}{\lfloor P \rfloor\ r\ (Q)}\ \text{SILseq}$$

$$\frac{P : \lceil P' \rceil \quad (P')\ r\ \lfloor Q' \rfloor \quad Q : \lfloor Q \rfloor}{(P)\ r\ \lfloor Q \rfloor}\ \text{ILseq}$$



Using standard braces

$$\frac{P : (\!|\, P' \,|\!)_{\gamma} \qquad (\!|\, P' \,|\!)_{\alpha}\ r\ (\!|\, Q' \,|\!)_{\beta} \qquad Q : (\!|\, Q' \,|\!)_{\delta}}{(\!|\, P \,|\!)_{\alpha \sqcup \gamma \sqcup \delta}\ r\ (\!|\, Q \,|\!)_{\beta \sqcup \bar{\gamma} \sqcup \delta}}\ [\text{Cons}]$$

A single rule (for all triples!): 256 cases!

A Logic for All Reasons

$$\frac{P : (\!| P' |\!)_\gamma \qquad (\!| P' |\!)_\alpha \; r \; (\!| Q' |\!)_\beta \qquad Q : (\!| Q' |\!)_\delta}{(\!| P |\!)_{\alpha \sqcup \gamma \sqcup \bar{\delta}} \; r \; (\!| Q |\!)_{\beta \sqcup \bar{\gamma} \sqcup \delta}} \text{ [Cons]}$$

$$\frac{\alpha \sqsubseteq \gamma \qquad (\!| P |\!)_\alpha \; r \; (\!| Q |\!)_\beta \qquad \beta \sqsubseteq \delta}{(\!| P |\!)_\gamma \; r \; (\!| Q |\!)_\delta} \text{ [Ord]}$$

$$\frac{(\!| P |\!)_\alpha \; r_1 \; (\!| R |\!)_\beta \quad (\!| R |\!)_\gamma \; r_2 \; (\!| Q |\!)_\delta}{(\!| P |\!)_{\alpha \sqcup \gamma} \; r_1 ; r_2 \; (\!| Q |\!)_{\beta \sqcup \delta}} \text{ [Seq]}$$

$$\frac{(\!| P |\!)_\alpha \; r \; (\!| Q |\!)_\beta \quad x \notin fv(r)}{(\!| \exists x \,.\, P |\!)_\alpha \; r \; (\!| \exists x \,.\, Q |\!)_\beta} \text{ [Exists]}$$

$$\frac{(\!| P_1 |\!)_\alpha \; r \; (\!| Q_1 |\!)_\beta \quad (\!| P_2 |\!)_\alpha \; r \; (\!| Q_2 |\!)_\beta}{(\!| P_1 \vee P_2 |\!)_\alpha \; r \; (\!| Q_1 \vee Q_2 |\!)_\beta} \text{ [Disj]}$$

$$\frac{(\!| P |\!)_\alpha \; r_1 \; (\!| Q |\!)_\beta \quad (\!| P |\!)_\gamma \; r_2 \; (\!| Q |\!)_\delta}{(\!| P |\!)_{\alpha \sqcup \gamma} \; r_1 + r_2 \; (\!| Q |\!)_{\beta \sqcup \delta}} \text{ [Choice]}$$

$$\frac{(\!| P |\!)_\alpha \; r_1 \; (\!| Q_1 |\!)_\beta \quad (\!| P |\!)_\gamma \; r_2 \; (\!| Q_2 |\!)_\delta}{(\!| P |\!)_{\alpha \sqcup \gamma \sqcup \lfloor \cdot \rfloor} \; r_1 + r_2 \; (\!| Q_1 \vee Q_2 |\!)_{\beta \sqcup \delta}} \text{ [ChoiceL]}$$

$$\frac{(\!| P |\!)_\alpha \; r \; (\!| P |\!)_\beta}{(\!| P |\!)_\alpha \; r^* \; (\!| P |\!)_\beta} \text{ [Iter]}$$

$$\frac{\forall n \geq 0 . (\!| P_n |\!)_\alpha \; r \; (\!| P_{n+1} |\!)_\beta}{(\!| P_0 |\!)_{\alpha \sqcup \lfloor \cdot \rfloor} \; r^* \; (\!| \vee_{n \geq 0} P_n |\!)_\beta} \text{ [IterL]}$$

Axioms Schema

$$P : \lceil R \rceil \iff R \subseteq P$$

$$P : \lfloor R \rfloor \iff P \subseteq R$$

$$P : |R| \iff R = P$$

$$P : (R) \iff \text{no relation}$$

Axioms Schema

$$\frac{Q : (\!| [\![r]\!] P |\!)_\alpha}{(P)\ r\ (\!| Q |\!)_\alpha}$$

Axioms for forward reasoning

$$P : \lceil R \rceil \iff R \subseteq P$$

$$P : \lfloor R \rfloor \iff P \subseteq R$$

$$P : |R| \iff R = P$$

$$P : (R) \iff \text{no relation}$$

Axioms Schema

$$\frac{Q : (\!| [\![r]\!] P |\!)_\alpha}{(P)\ r\ (\!| Q |\!)_\alpha}$$

Axioms for forward reasoning

$$P : \lceil R \rceil \iff R \subseteq P$$

$$P : \lfloor R \rfloor \iff P \subseteq R$$

$$P : |R| \iff R = P$$

$$P : (R) \iff \text{no relation}$$

$$\frac{P \wedge b : |\, [\![b?]\!] P \,|}{(P)\ b?\ |P \wedge b|}$$

Axioms Schema

$$\frac{Q : (\!| [\![r]\!] P |\!)_\alpha}{(P)\ r\ (\!| Q |\!)_\alpha}$$

Axioms for forward reasoning

$$P : \lceil R \rceil \iff R \subseteq P$$

$$P : \lfloor R \rfloor \iff P \subseteq R$$

$$P : |R| \iff R = P$$

$$P : (R) \iff \text{no relation}$$

$$\frac{P : (\!| [\![\overleftarrow{r}]\!] Q |\!)_\alpha}{(\!| P |\!)_\alpha\ r\ (Q)}$$

Axioms for backward reasoning

$$\frac{P \wedge b : |\, [\![b?]\!] P \,|}{(P)\ b?\ |P \wedge b|}$$

Axioms Schema

$$\frac{Q : (\!| \, [\![r]\!] P \, |\!)_\alpha}{(P)\ r\ (\!| \, Q \, |\!)_\alpha}$$

Axioms for forward reasoning

$$P : \lceil R \rceil \iff R \subseteq P$$
$$P : \lfloor R \rfloor \iff P \subseteq R$$
$$P : |R| \iff R = P$$
$$P : (R) \iff \text{no relation}$$

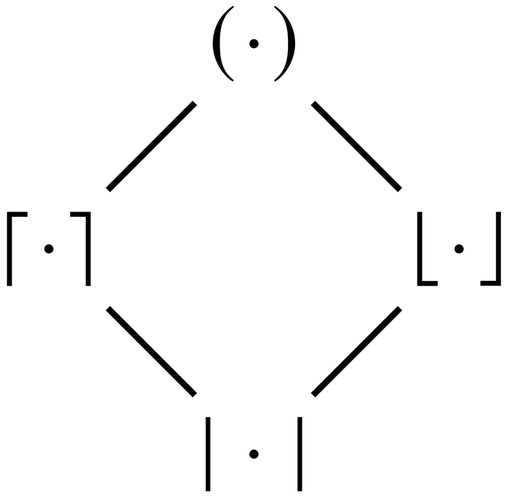
$$\frac{P : (\!| \, [\![\overleftarrow{r}]\!] Q \, |\!)_\alpha}{(\!| \, P \, |\!)_\alpha\ r\ (Q)}$$

Axioms for backward reasoning

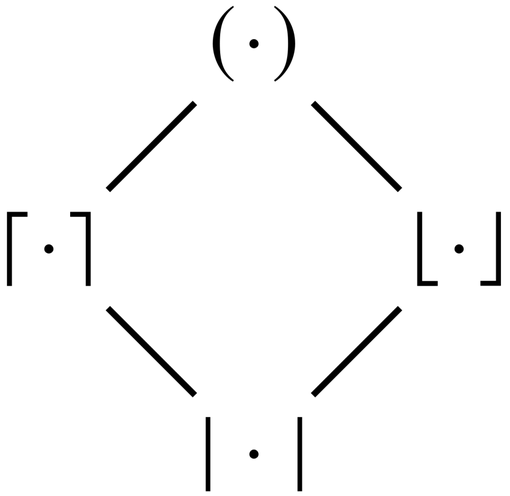
$$\frac{P \wedge b : |\, [\![b?]\!] P \,|}{(P)\ b?\ |P \wedge b|}$$

$$\frac{Q \wedge b : |\, [\![\overleftarrow{b?}]\!] Q \,|}{|Q \wedge b|\ b?\ (Q)}$$

The whole is more than the sum of the parts



The whole is more than the sum of the parts

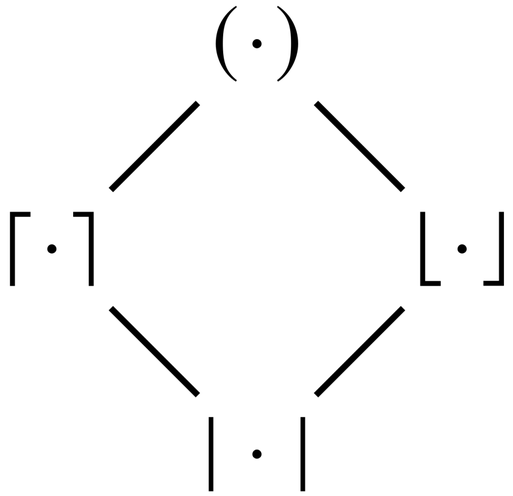


$$\frac{(\!|\, P \,|\!)_{\alpha}\ r\ (\!|\, Q \,|\!)_{\beta} \quad (\!|\, P \,|\!)_{\gamma}\ r\ (\!|\, Q \,|\!)_{\delta}}{(\!|\, P \,|\!)_{\alpha \sqcap \gamma}\ r\ (\!|\, Q \,|\!)_{\beta \sqcap \delta}}$$

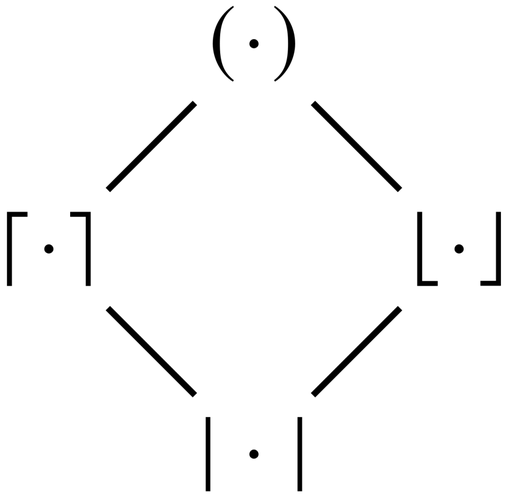
The whole is more than the sum of the parts

Same P,Q but different approximations

$$\frac{(\!|\, P \,|\!)_{\alpha}\ r\ (\!|\, Q \,|\!)_{\beta} \quad (\!|\, P \,|\!)_{\gamma}\ r\ (\!|\, Q \,|\!)_{\delta}}{(\!|\, P \,|\!)_{\alpha \sqcap \gamma}\ r\ (\!|\, Q \,|\!)_{\beta \sqcap \delta}}$$



The whole is more than the sum of the parts

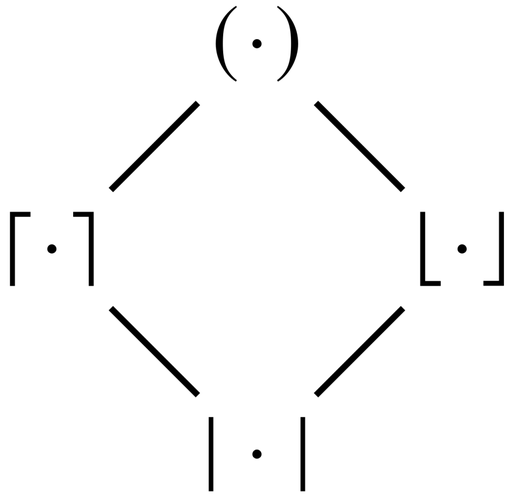


Same P,Q but different approximations

$$\frac{(\!|\, P \,|\!)_{\alpha}\ r\ (\!|\, Q \,|\!)_{\beta} \quad (\!|\, P \,|\!)_{\gamma}\ r\ (\!|\, Q \,|\!)_{\delta}}{(\!|\, P \,|\!)_{\alpha \sqcap \gamma}\ r\ (\!|\, Q \,|\!)_{\beta \sqcap \delta}}$$

We take the most general approximation

The whole is more than the sum of the parts



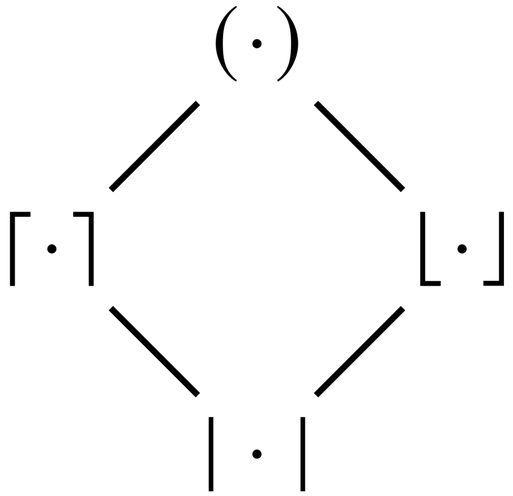
Same P,Q but different approximations

$$\frac{(\!| P |\!)_{\alpha}\ r\ (\!| Q |\!)_{\beta} \quad (\!| P |\!)_{\gamma}\ r\ (\!| Q |\!)_{\delta}}{(\!| P |\!)_{\alpha \sqcap \gamma}\ r\ (\!| Q |\!)_{\beta \sqcap \delta}}$$

We take the most general approximation

$$\frac{P \wedge b : |\, [\![b?]\!](P \wedge b) \,|}{(P \wedge b)\ b?\ |P \wedge b|} \qquad \frac{P \wedge b : |\, [\![\overleftarrow{b?}]\!](P \wedge b) \,|}{|P \wedge b|\ b?\ (P \wedge b)}$$

The whole is more than the sum of the parts



Same P,Q but different approximations

$$\frac{(\!| P |\!)_{\alpha}\ r\ (\!| Q |\!)_{\beta} \quad (\!| P |\!)_{\gamma}\ r\ (\!| Q |\!)_{\delta}}{(\!| P |\!)_{\alpha \sqcap \gamma}\ r\ (\!| Q |\!)_{\beta \sqcap \delta}}$$

We take the most general approximation

$$\frac{P \wedge b : |[\![b?]\!](P \wedge b)| \qquad P \wedge b : |[\![\overleftarrow{b?}]\!](P \wedge b)|}{(P \wedge b)\ b?\ |P \wedge b| \qquad |P \wedge b|\ b?\ (P \wedge b)}$$

$$\frac{}{|P \wedge b|\ b?\ |P \wedge b|}$$

Axioms Schema

$$\frac{Q : (\!| [\![r]\!] P |\!)_\alpha}{(P)\ r\ (\!| Q |\!)_\alpha}$$

Sets axioms for forward semantics

$$P : \lceil R \rceil \iff R \subseteq P$$

$$P : \lfloor R \rfloor \iff P \subseteq R$$

$$P : |R| \iff R = P$$

$$P : (R) \iff \text{no relation}$$

$$\frac{P : (\!| [\![\overleftarrow{r}]\!] Q |\!)_\alpha}{(\!| P |\!)_\alpha\ r\ (Q)}$$

Sets axioms for backward semantics

Axioms Schema

$$\frac{Q : (\!| [\![r]\!] P |\!)_\alpha}{(P)\ r\ (\!| Q |\!)_\alpha}$$

Sets axioms for forward semantics

$$P : \lceil R \rceil \iff R \subseteq P$$
$$P : \lfloor R \rfloor \iff P \subseteq R$$
$$P : |R| \iff R = P$$
$$P : (R) \iff \text{no relation}$$

$$\frac{P : (\!| [\![\overleftarrow{r}]\!] Q |\!)_\alpha}{(\!| P |\!)_\alpha\ r\ (Q)}$$

Sets axioms for backward semantics

Floyd's style

$$\frac{\exists n\,.\,P[n/x] \wedge x = e[n/x] : |\,[\![x := e]\!] P\,|}{(P)\ x := e\ |\exists n\,.\,P[n/x] \wedge x = e[n/x]|}$$

Axioms Schema

$$\frac{Q : (\!| \, [\![r]\!] P \, |\!)_\alpha}{(P)\ r\ (\!| \, Q \, |\!)_\alpha}$$

Sets axioms for forward semantics

$$P : \lceil R \rceil \iff R \subseteq P$$
$$P : \lfloor R \rfloor \iff P \subseteq R$$
$$P : |R| \iff R = P$$
$$P : (R) \iff \text{no relation}$$

$$\frac{P : (\!| \, [\![\overleftarrow{r}]\!] Q \, |\!)_\alpha}{(\!| \, P \, |\!)_\alpha\ r\ (Q)}$$

Sets axioms for backward semantics

Floyd's style

$$\frac{\exists n \,.\, P[n/x] \wedge x = e[n/x] : |\, [\![x := e]\!] P \,|}{(P)\ x := e\ |\exists n \,.\, P[n/x] \wedge x = e[n/x]|}$$

Hoare's style

$$\frac{Q[e/x] : |\, [\![\overleftarrow{x := e}]\!] Q \,|}{|Q[e/x]|\ x := e\ (Q)}$$

The whole is more than the sum of the parts

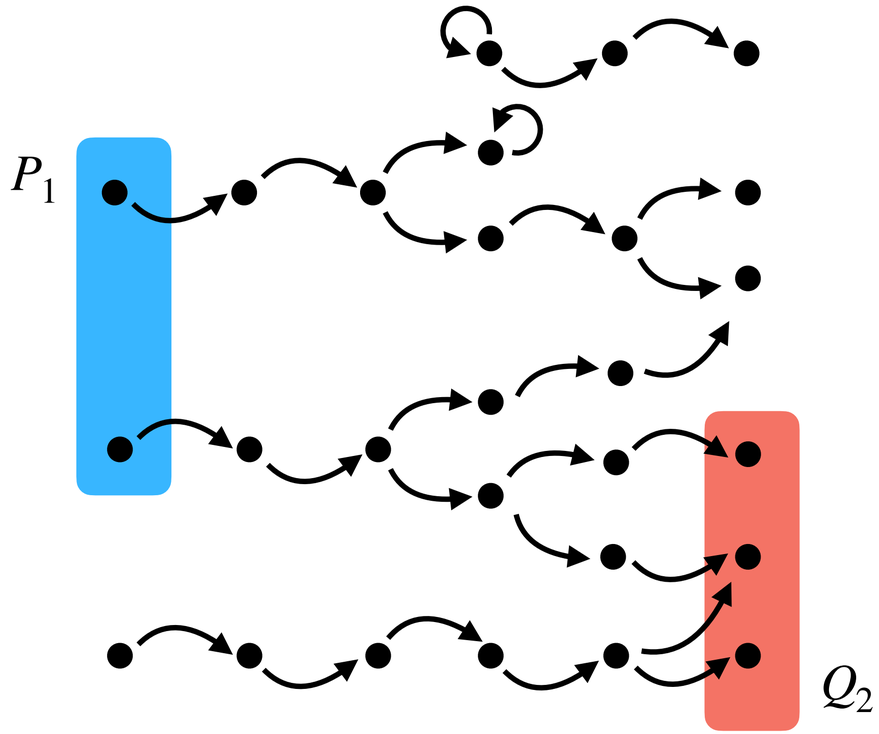
$$\frac{(\!|\, P_1 \,|\!)_\alpha \; r \; |\, Q_1 \,| \qquad |\, P_2 \,| \; r \; (\!|\, Q_2 \,|\!)_\beta}{\lfloor P_1 \wedge P_2 \rfloor \; r \; \lfloor Q_1 \wedge Q_2 \rfloor}$$

- ➡ Every state in $Q_1 \wedge Q_2$ is reachable
- ➡ Every state in $P_1 \wedge P_2$ is guaranteed to reach a state in $Q_1 \wedge Q_2$
- ➡ If $Q_1 \wedge Q_2$ are errors, $P_1 \wedge P_2$ are some source of errors

The whole is more than the sum of the parts

$$\frac{(\!|\, P_1 \,|\!)_\alpha \; r \; |\, Q_1 \,| \qquad |\, P_2 \,| \; r \; (\!|\, Q_2 \,|\!)_\beta}{\lfloor P_1 \wedge P_2 \rfloor \; r \; \lfloor Q_1 \wedge Q_2 \rfloor}$$

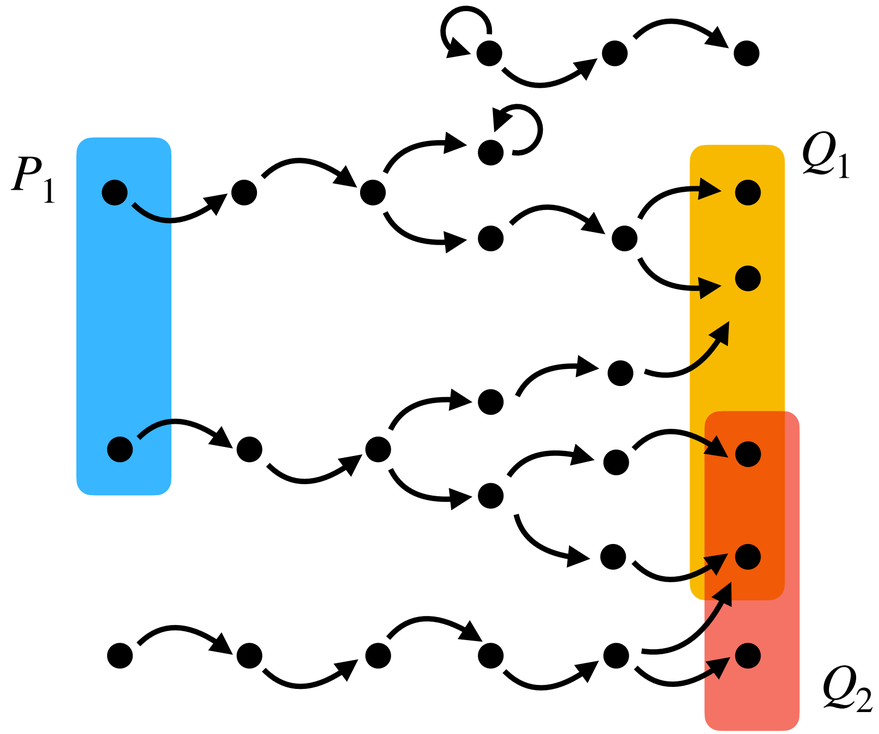
- ➡ Every state in $Q_1 \wedge Q_2$ is reachable
- ➡ Every state in $P_1 \wedge P_2$ is guaranteed to reach a state in $Q_1 \wedge Q_2$
- ➡ If $Q_1 \wedge Q_2$ are errors, $P_1 \wedge P_2$ are some source of errors



The whole is more than the sum of the parts

$$\frac{(\!|\, P_1 \,|\!)_\alpha \; r \; |\, Q_1 \,| \qquad |\, P_2 \,| \; r \; (\!|\, Q_2 \,|\!)_\beta}{\lfloor P_1 \wedge P_2 \rfloor \; r \; \lfloor Q_1 \wedge Q_2 \rfloor}$$

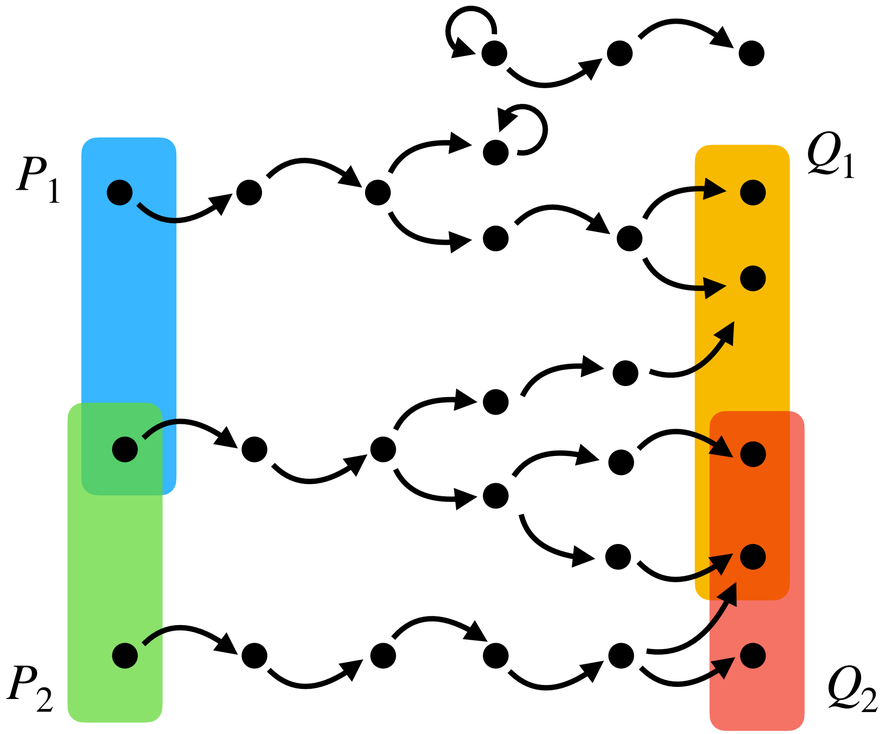
- ➡ Every state in $Q_1 \wedge Q_2$ is reachable
- ➡ Every state in $P_1 \wedge P_2$ is guaranteed to reach a state in $Q_1 \wedge Q_2$
- ➡ If $Q_1 \wedge Q_2$ are errors, $P_1 \wedge P_2$ are some source of errors



The whole is more than the sum of the parts

$$\frac{(\!|\, P_1 \,|\!)_\alpha \; r \; |\, Q_1 \,| \qquad |\, P_2 \,| \; r \; (\!|\, Q_2 \,|\!)_\beta}{\lfloor P_1 \wedge P_2 \rfloor \; r \; \lfloor Q_1 \wedge Q_2 \rfloor}$$

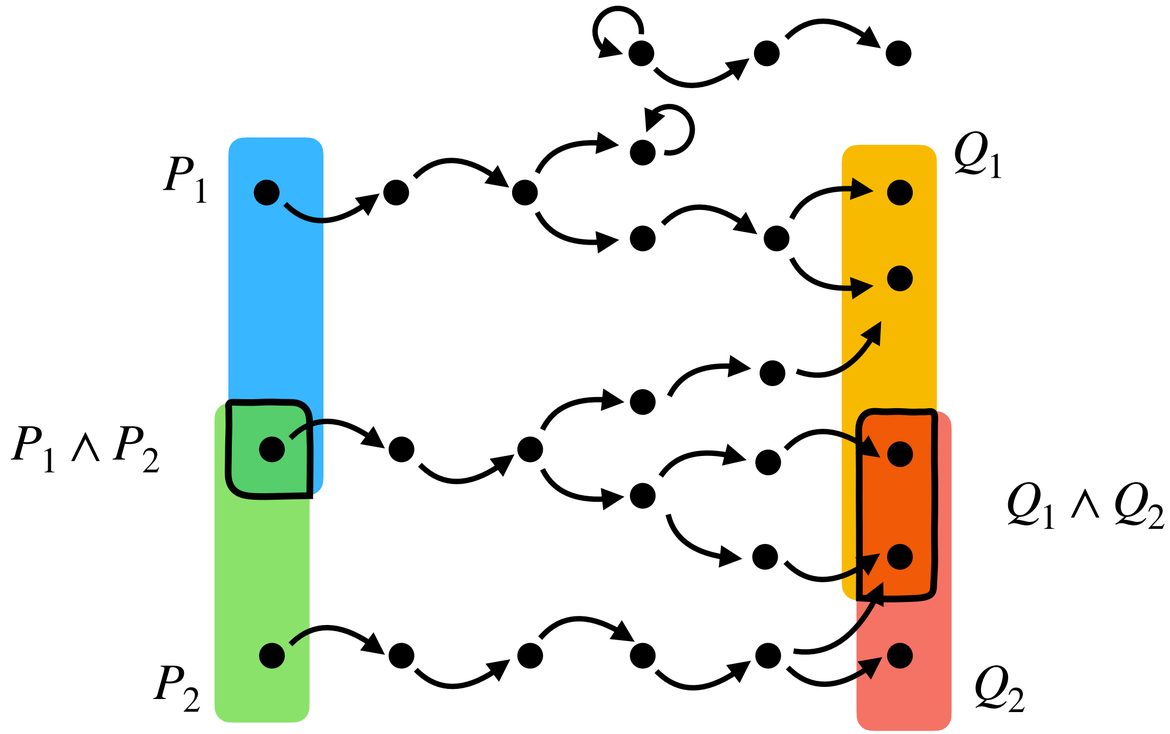
- ➡ Every state in $Q_1 \wedge Q_2$ is reachable
- ➡ Every state in $P_1 \wedge P_2$ is guaranteed to reach a state in $Q_1 \wedge Q_2$
- ➡ If $Q_1 \wedge Q_2$ are errors, $P_1 \wedge P_2$ are some source of errors



The whole is more than the sum of the parts

$$\frac{(\!|\, P_1 \,|\!)_\alpha \; r \; |\, Q_1 \,| \qquad |\, P_2 \,| \; r \; (\!|\, Q_2 \,|\!)_\beta}{\lfloor P_1 \wedge P_2 \rfloor \; r \; \lfloor Q_1 \wedge Q_2 \rfloor}$$

- ➡ Every state in $Q_1 \wedge Q_2$ is reachable
- ➡ Every state in $P_1 \wedge P_2$ is guaranteed to reach a state in $Q_1 \wedge Q_2$
- ➡ If $Q_1 \wedge Q_2$ are errors, $P_1 \wedge P_2$ are some source of errors



Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Over

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

Floyd's axiom validity for IL

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

Floyd's axiom validity for IL

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

Floyd's axiom validity for IL

We apply the backward semantics to the postcondition S

$$S[e/x]$$
$$\equiv (\exists n \,.\, P[n/x] \wedge x = e[n/x])[e/x]$$

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

Floyd's axiom validity for IL

We apply the backward semantics to the postcondition S

$$\begin{aligned} & S[e/x] \\ \equiv\ & (\exists n \,.\, P[n/x] \wedge x = e[n/x])[e/x] \\ \equiv\ & (\exists n \,.\, P[n/x] \wedge e = e[n/x]) \end{aligned}$$

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

Floyd's axiom validity for IL

We apply the backward semantics to the postcondition S

$$\begin{aligned} & S[e/x] \\ \equiv\ & (\exists n \,.\, P[n/x] \wedge x = e[n/x])[e/x] \\ \equiv\ & (\exists n \,.\, P[n/x] \wedge e = e[n/x]) \\ \Leftarrow\ & (\exists n \,.\, P[n/x] \wedge e = e[n/x] \wedge x = n) \end{aligned}$$

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

Floyd's axiom validity for IL

We apply the backward semantics to the postcondition S

$$\begin{aligned}
& S[e/x] \\
\equiv\ & (\exists n \,.\, P[n/x] \wedge x = e[n/x])[e/x] \\
\equiv\ & (\exists n \,.\, P[n/x] \wedge e = e[n/x]) \\
\Leftarrow\ & (\exists n \,.\, P[n/x] \wedge e = e[n/x] \wedge x = n) \\
\equiv\ & (\exists n \,.\, P[n/x] \wedge x = n)
\end{aligned}$$

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

Floyd's axiom validity for IL

We apply the backward semantics to the postcondition S

$$\begin{aligned}
& S[e/x] \\
\equiv\ & (\exists n \,.\, P[n/x] \wedge x = e[n/x])[e/x] \\
\equiv\ & (\exists n \,.\, P[n/x] \wedge e = e[n/x]) \\
\Leftarrow\ & (\exists n \,.\, P[n/x] \wedge e = e[n/x] \wedge x = n) \\
\equiv\ & (\exists n \,.\, P[n/x] \wedge x = n) \\
\equiv\ & P
\end{aligned}$$

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

We apply the backward semantics to the postcondition S

$$P \Rightarrow S[e/x]$$

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

Floyd's axiom validity for IL

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

$$\frac{P : \lfloor [\![\overleftarrow{x := e}]\!]S \rfloor}{\lfloor P \rfloor\ x := e\ (S)}$$

We apply the backward semantics to the postcondition S

$$P \Rightarrow S[e/x]$$

Floyd's axiom validity for IL

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

We apply the backward semantics to the postcondition S

$$P \Rightarrow S[e/x]$$

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

$$\frac{P : \lfloor [\![\overleftarrow{x := e}]\!]S \rfloor}{\lfloor P \rfloor\ x := e\ (S)}$$

Floyd's axiom validity for IL

Floyd's axiom validity for SIL

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

Floyd's axiom not valid for NC

We apply the backward semantics to the postcondition S

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

$$\frac{P : \lfloor [\![\overleftarrow{x := e}]\!]S \rfloor}{\lfloor P \rfloor\ x := e\ (S)}$$

$$P \Rightarrow S[e/x]$$

Floyd's axiom validity for IL

Floyd's axiom validity for SIL

Recover Floyd's axiom validity

$$[\![x := e]\!]P = \underbrace{\exists n \,.\, P[n/x] \wedge x = e[n/x]}_{S}$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Floyd's axiom validity for HL

Over

$$\frac{S : \lceil [\![x := e]\!]P \rceil}{(P)\ x := e\ \lceil S \rceil}$$

Floyd's axiom not valid for NC

We apply the backward semantics to the postcondition S

$$P \Rightarrow S[e/x]$$

Under

$$\frac{S : \lfloor [\![x := e]\!]P \rfloor}{(P)\ x := e\ \lfloor S \rfloor}$$

$$\frac{P : \lfloor [\![\overleftarrow{x := e}]\!]S \rfloor}{\lfloor P \rfloor\ x := e\ (S)}$$

Floyd's axiom validity for IL

Floyd's axiom validity for SIL

Combined axiom

$$\lfloor P \rfloor\ x := e\ |S|$$

Recover Hoare's axiom validity

$$[\![x := e]\!]P = \exists n \,.\, P[n/x] \wedge x = e[n/x]$$

$$[\![\overleftarrow{x := e}]\!]Q = Q[e/x]$$

Hoare's axiom validity for HL

Hoare's axiom validity for NC

Over

$$\frac{Q : \lceil [\![x := e]\!] Q[e/x] \rceil}{(Q[e/x])\ x := e\ \lceil Q \rceil}$$

$$\frac{Q[e/x] : \lfloor [\![x := e]\!] Q \rfloor}{(Q[e/x])\ x := e\ \lfloor Q \rfloor}$$

Under

Hoare’s axiom not valid for IL

$$\frac{Q[e/x] : \lfloor [\![\overleftarrow{x := e}]\!] Q \rfloor}{\lfloor Q[e/x] \rfloor\ x := e\ (Q)}$$

Hoare's axiom validity for SIL

Combined axiom

$$|\,Q[e/x]\,|\ x := e\ \lceil Q \rceil$$

Non-termination and reachability

- A program r **must-diverge** on input P if $[\![\mathrm{r}]\!]P = \varnothing$
  - in our notation, e.g., $(P)\ r\ \texttt{|false|}$

Non-termination and reachability

- A program r **must-diverge** on input P if $[\![\mathrm{r}]\!]P = \varnothing$
  - in our notation, e.g., $(P)\ r\ |\texttt{false}|$
- [OOPSLA24] Unter logic triples $[P]\ r\ [\infty]$ mean that r **may-diverge** on any state in P
  - extending our notation: $\lfloor P \rfloor\ r\ (\texttt{false})^{\infty}$

Non-termination and reachability

- A program r **must-diverge** on input P if $[\![\mathrm{r}]\!]P = \varnothing$
  - in our notation, e.g., $(P)\ r\ |\texttt{false}|$
- [OOPSLA24] Unter logic triples $[P]\ r\ [\infty]$ mean that r **may-diverge** on any state in P
  - extending our notation: $\lfloor P \rfloor\ r\ (\texttt{false})^{\infty}$
- By duality: **unreachability triples**
  - extending our notation: $(\texttt{false})^{\infty}\ r\ \lfloor P \rfloor$

Non-termination and reachability

- A program r **must-diverge** on input P if $[\![\mathrm{r}]\!]P = \varnothing$
  - in our notation, e.g., $(P)\ r\ |\texttt{false}|$
- [OOPSLA24] Unter logic triples $[P]\ r\ [\infty]$ mean that r **may-diverge** on any state in P
  - extending our notation: $\lfloor P \rfloor\ r\ (\texttt{false})^{\infty}$
- By duality: **unreachability triples**
  - extending our notation: $(\texttt{false})^{\infty}\ r\ \lfloor P \rfloor$
- 64 triples are now possible... still to be fully investigated

Non-termination analysis

Let $[P]\ r\ [\infty]$ assert that r **may** diverge from **every state** in P

Non-termination analysis

Let $[P]\ r\ [\infty]$ assert that r **may** diverge from **every state** in P

$$\frac{[P_1]\ r\ [\infty] \quad [P_2]\ r\ [\infty]}{[P_1 \vee P_2]\ r\ [\infty]}$$

Non-termination analysis

Let $[P]\ r\ [\infty]$ assert that r **may** diverge from **every state** in P

$$\frac{[P_1]\ r\ [\infty] \quad [P_2]\ r\ [\infty]}{[P_1 \vee P_2]\ r\ [\infty]}$$

$$\frac{[P \wedge b]\ r\ [\infty]}{[P \wedge b]\ \text{while } b \text{ do } r\ [\infty]}$$

Which rule to use?

can my loop diverge?

$$\frac{[P \wedge b]\ r\ [\infty]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

$$\frac{\{P \wedge b\}\ r\ \{P \wedge b\}}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

HL

1

$$\frac{[P \wedge b]\ r\ [P \wedge b]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

IL

2

$$\frac{\langle P \wedge b\rangle\ r\ \langle P \wedge b\rangle}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

SIL

3

Which rule to use?

$$\frac{[P \wedge b]\ r\ [\infty]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

$$\frac{\{P \wedge b\}\ r\ \{P \wedge b\}}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

HL

1

$$\frac{[P \wedge b]\ r\ [P \wedge b]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

IL

2

$$\frac{\langle P \wedge b\rangle\ r\ \langle P \wedge b\rangle}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

SIL

3

$$\text{while}\ (x > 0)\ \text{do}\ \{\ x := x + 1\ \}$$

Which rule to use?

can my loop diverge?

$$\frac{[P \wedge b]\ r\ [\infty]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

$$\frac{\{P \wedge b\}\ r\ \{P \wedge b\}}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

HL

1

$$\frac{[P \wedge b]\ r\ [P \wedge b]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

IL

2

$$\frac{\langle P \wedge b\rangle\ r\ \langle P \wedge b\rangle}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

SIL

3

$$\text{while}\ (x > 0)\ \text{do}\ \{\ x := x + 1 \oplus x := 0\ \}$$

Which rule to use?

can my loop diverge?

$$\frac{[P \wedge b]\ r\ [\infty]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

$$\frac{\{P \wedge b\}\ r\ \{P \wedge b\}}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

HL

1

not general enough!
(for nondeterministic programs)

while (r > 0) do {
1 := 0
}

$$\frac{[P \wedge b]\ r\ [P \wedge b]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

IL

2

$$\frac{\langle P \wedge b\rangle\ r\ \langle P \wedge b\rangle}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

SIL

3

Which rule to use?

can my loop diverge?

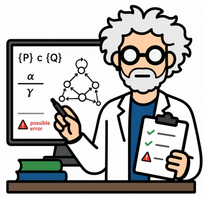
$$\frac{[P \wedge b]\ r\ [\infty]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

$$\frac{\{P \wedge b\}\ r\ \{P \wedge b\}}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

HL

1

not general enough!
(for nondeterministic programs)



$$\frac{[P \wedge b]\ r\ [P \wedge b]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

IL

2

$$\frac{\langle P \wedge b\rangle\ r\ \langle P \wedge b\rangle}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

SIL

3

while $(x > 0)$ do {
  $x := x - 1$
}

Which rule to use?

can my loop diverge?

$$\frac{[P \wedge b]\ r\ [\infty]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

$$\frac{\{P \wedge b\}\ r\ \{P \wedge b\}}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

HL

1

not general enough!
(for nondeterministic programs)

while (r > 0) do {
  1 := 0
}

$$\frac{[P \wedge b]\ r\ [P \wedge b]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

IL

2

UNSOUND!

while (r > 0) do {
  1 := 0
}

$$\frac{\langle P \wedge b\rangle\ r\ \langle P \wedge b\rangle}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

SIL

3

Which rule to use?

can my loop diverge?

$$\frac{[P \wedge b]\ r\ [\infty]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

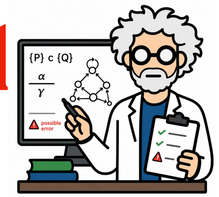
$$\frac{\{P \wedge b\}\ r\ \{P \wedge b\}}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

HL

1

not general enough!
(for nondeterministic programs)

while (r > 0) do {
  1 := 0
}



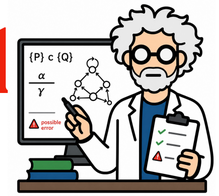
$$\frac{[P \wedge b]\ r\ [P \wedge b]}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

IL

2

UNSOUND!

while (r > 0) do {
  1 := 0
}

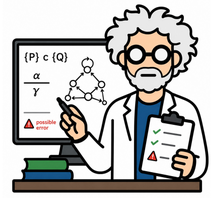


$$\frac{\langle P \wedge b\rangle\ r\ \langle P \wedge b\rangle}{[P \wedge b]\ \text{while}\ b\ \text{do}\ r\ [\infty]}$$

SIL

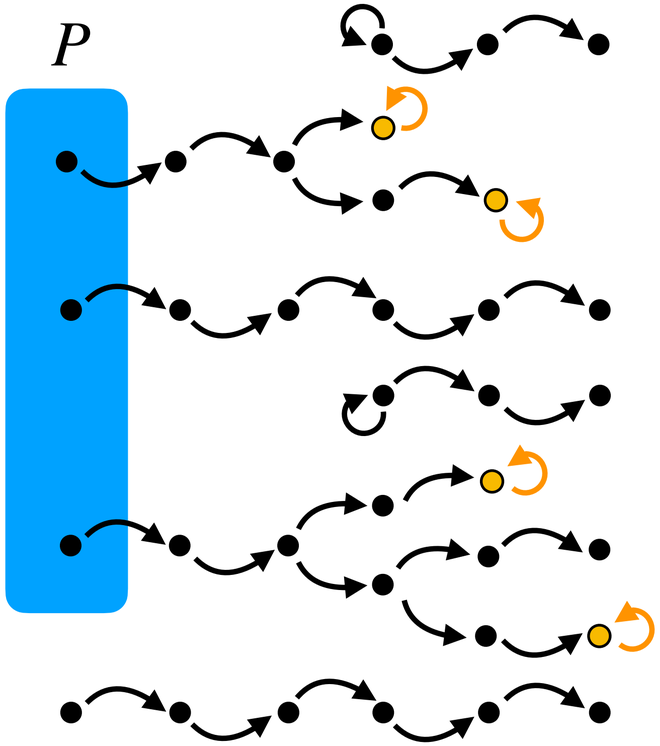
3

SOUNDS GOOD!



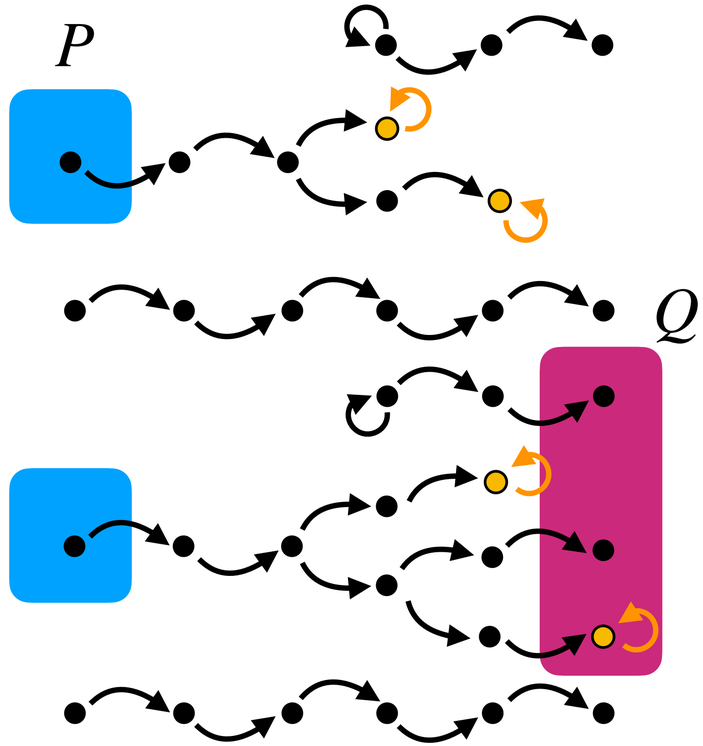
[OOPSLA24]
UnTer logic

Liberality



$\lceil P \rceil\ r\,(\texttt{false})^{\infty}$

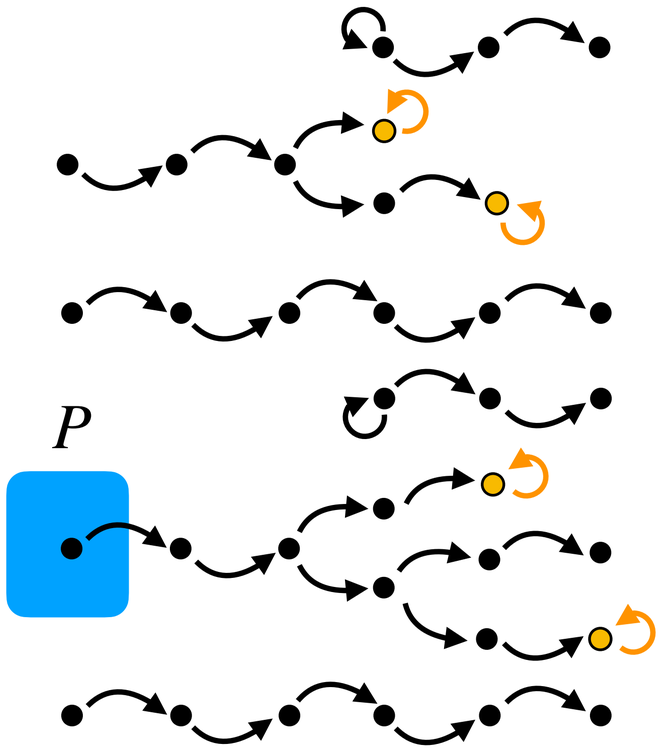
P is a necessary condition for some diverging execution



$\lfloor P \rfloor\ r\,(Q)^{\infty}$

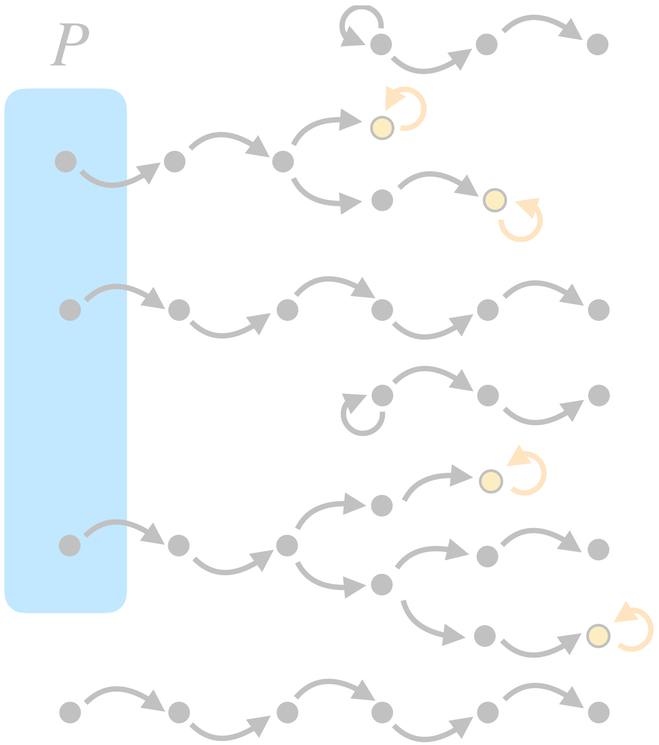
angelic partial correctness

Liberality



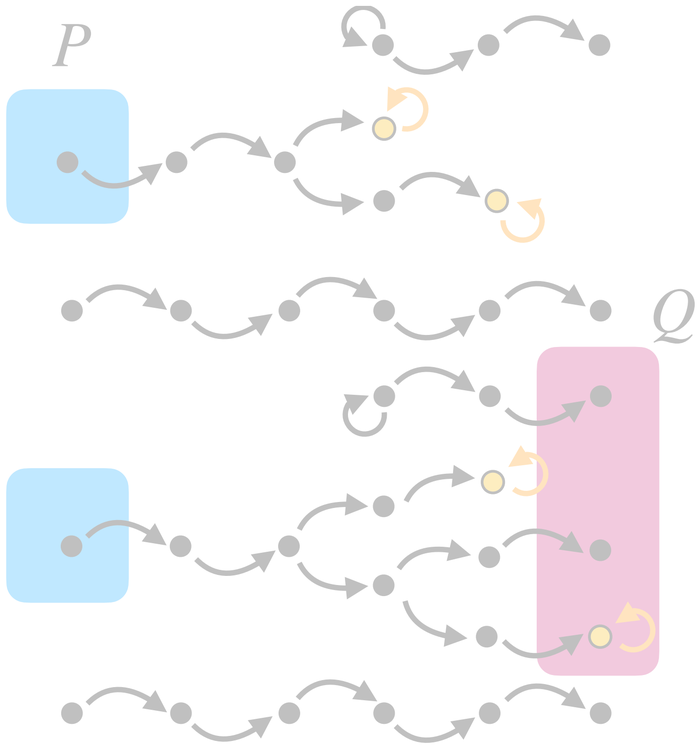
$\lfloor P \rfloor\ r\,(\texttt{false})^{\infty}$

every state in P has at least one diverging execution



$\lceil P \rceil\ r\,(\texttt{false})^{\infty}$

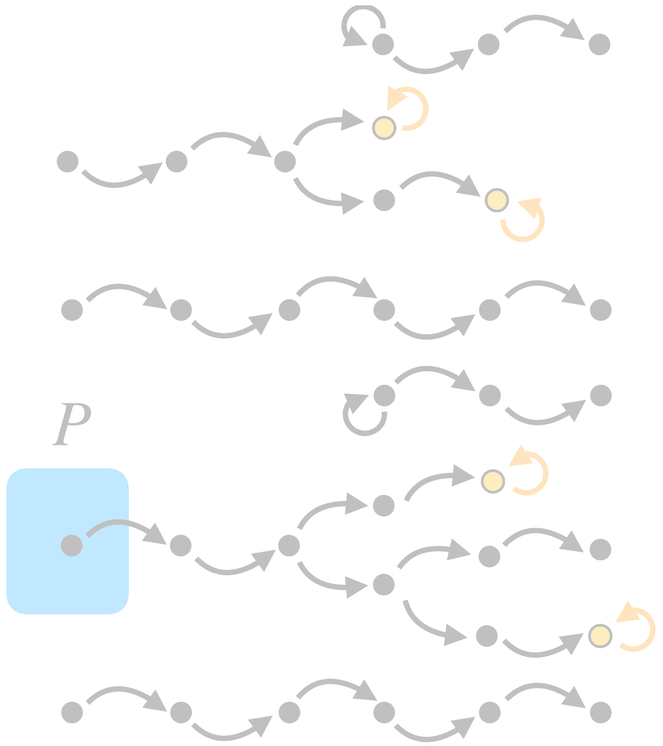
P is a necessary condition for some diverging execution



$\lfloor P \rfloor\ r\,(Q)^{\infty}$

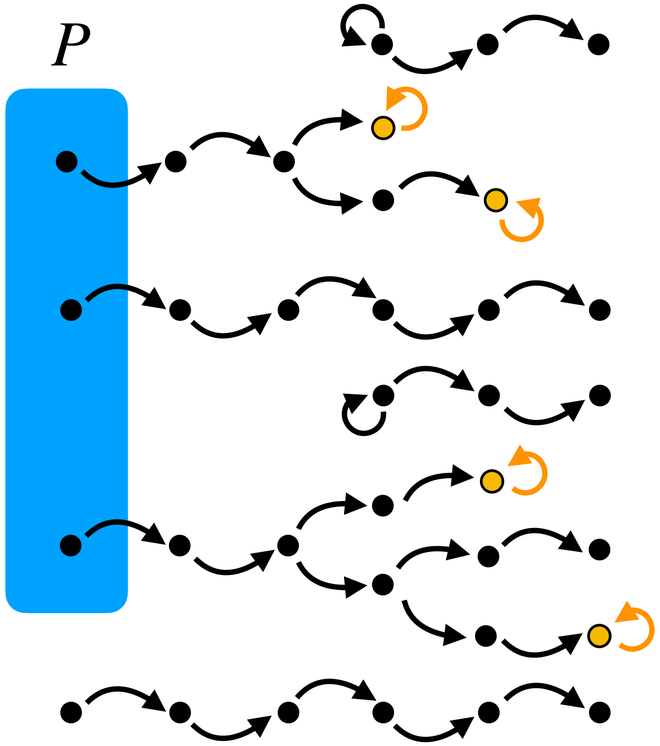
angelic partial correctness

Liberality



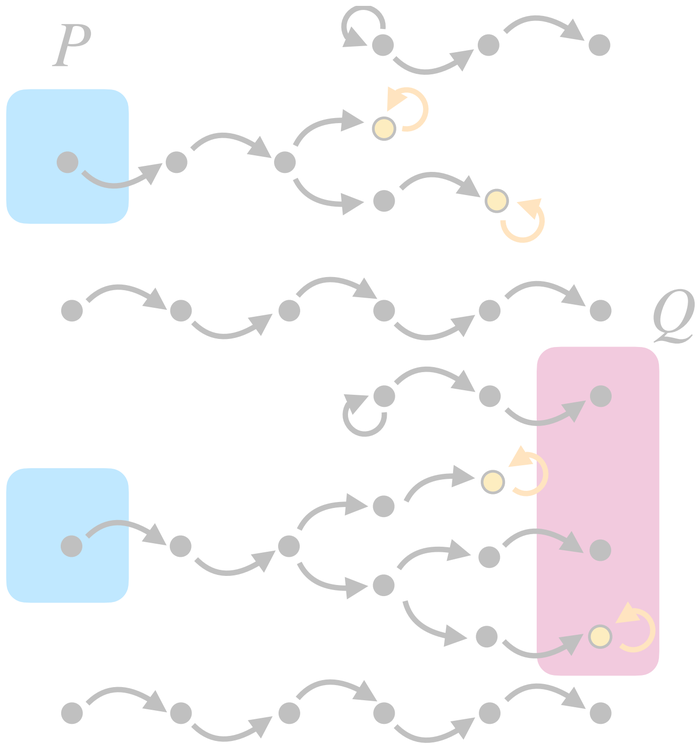
$\lfloor P \rfloor\ r\ (\texttt{false})^{\infty}$

every state in P has at least one diverging execution



$\lceil P \rceil\ r\ (\texttt{false})^{\infty}$

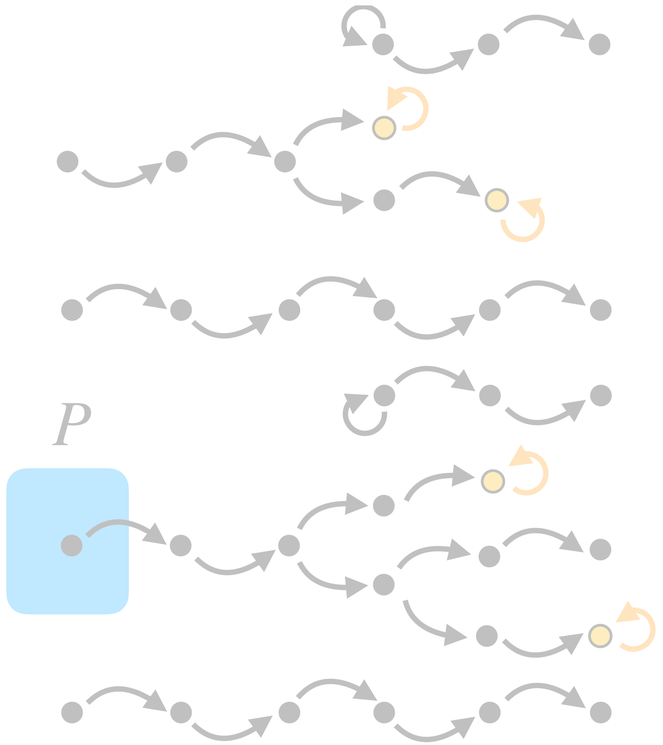
P is a necessary condition for some diverging execution



$\lfloor P \rfloor\ r\ (Q)^{\infty}$

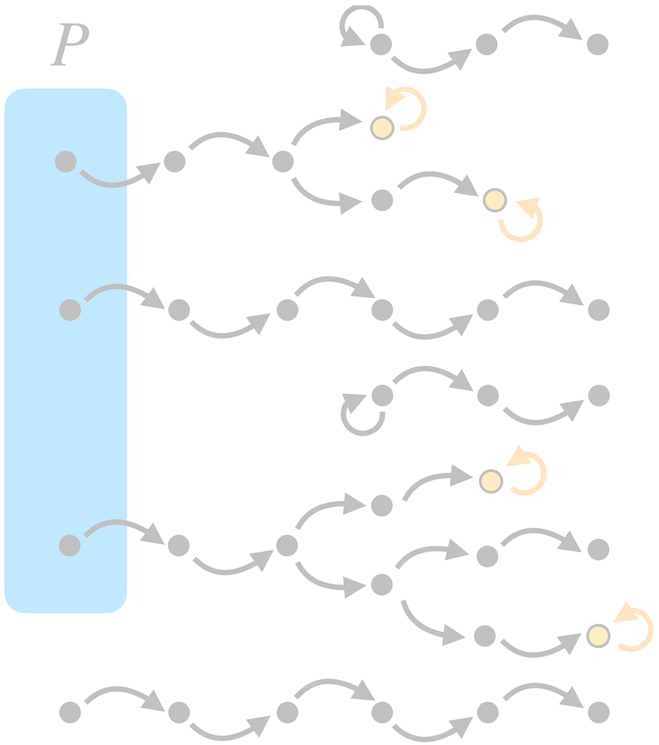
angelic partial correctness

Liberality



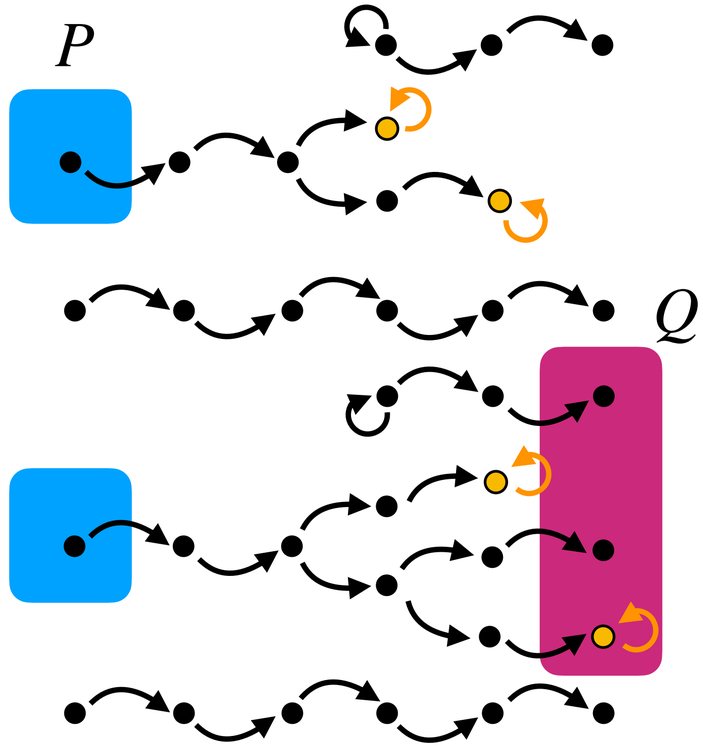
$\lfloor P \rfloor\ r\,(\texttt{false})^{\infty}$

every state in P has at least one diverging execution



$\lceil P \rceil\ r\,(\texttt{false})^{\infty}$

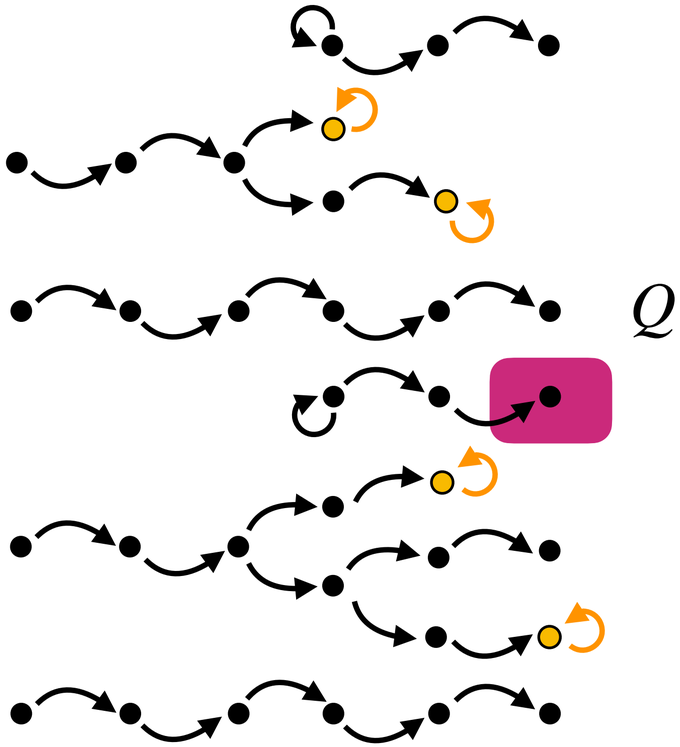
P is a necessary condition for some diverging execution



$\lfloor P \rfloor\ r\,(Q)^{\infty}$

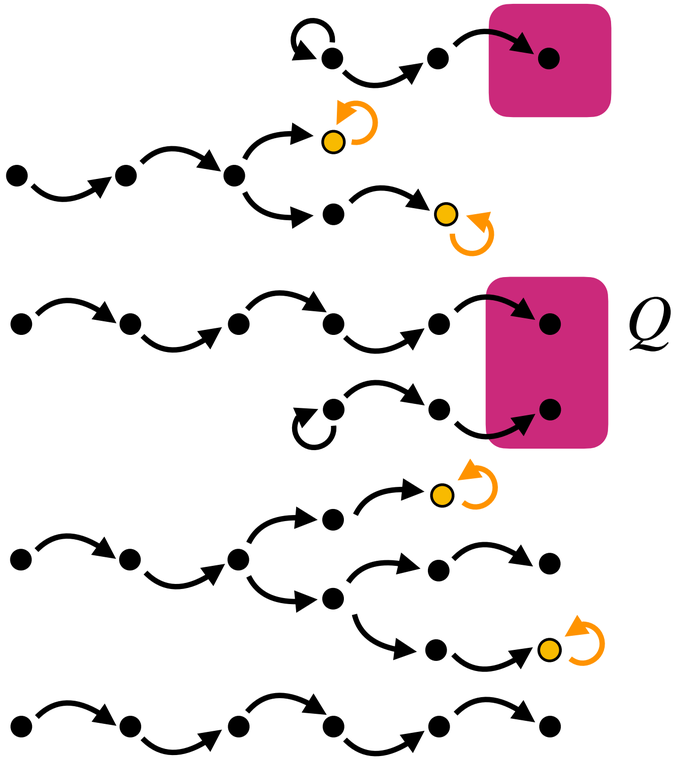
angelic partial correctness

By duality: (un)reachability



$(\texttt{false})^{\infty}\ r\ \lfloor Q \rfloor$

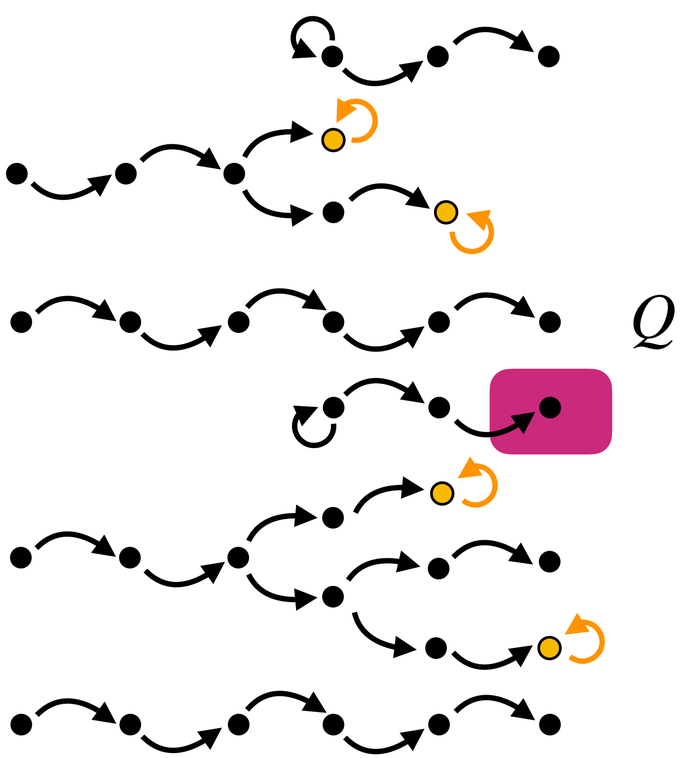
every state in Q
is unreachable



$(\texttt{false})^{\infty}\ r\ \lceil Q \rceil$

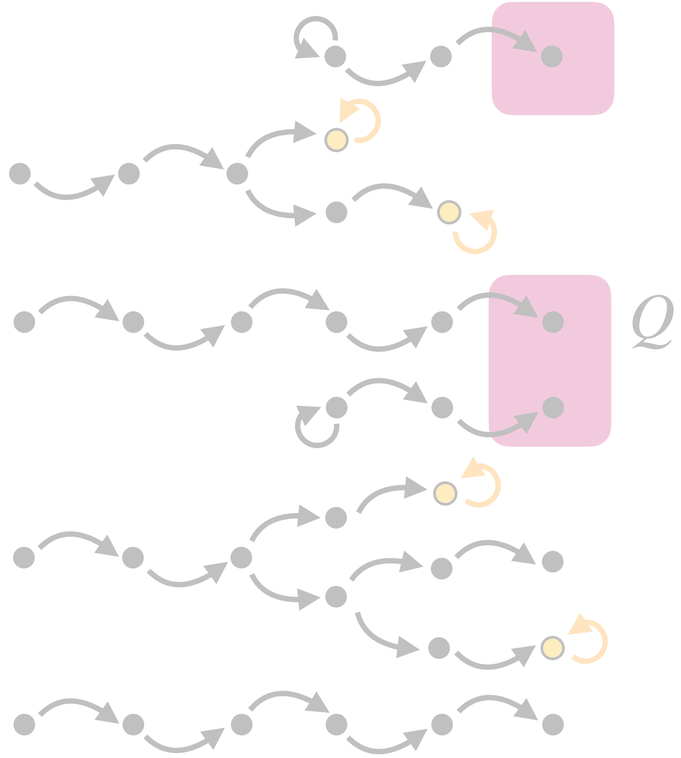
States outside Q
are reachable

By duality: (un)reachability



$(\texttt{false})^{\infty}\ r\ \lfloor Q \rfloor$

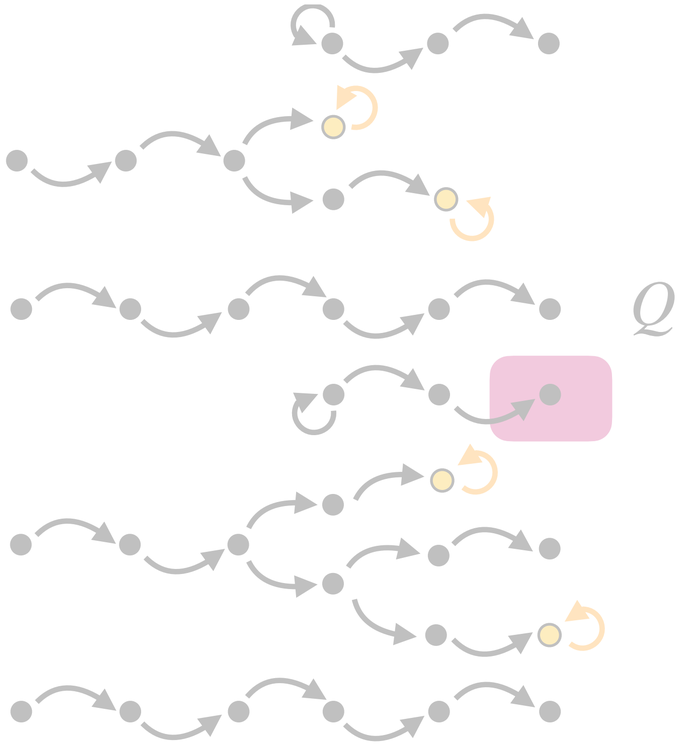
every state in Q
is unreachable



$(\texttt{false})^{\infty}\ r\ \lceil Q \rceil$

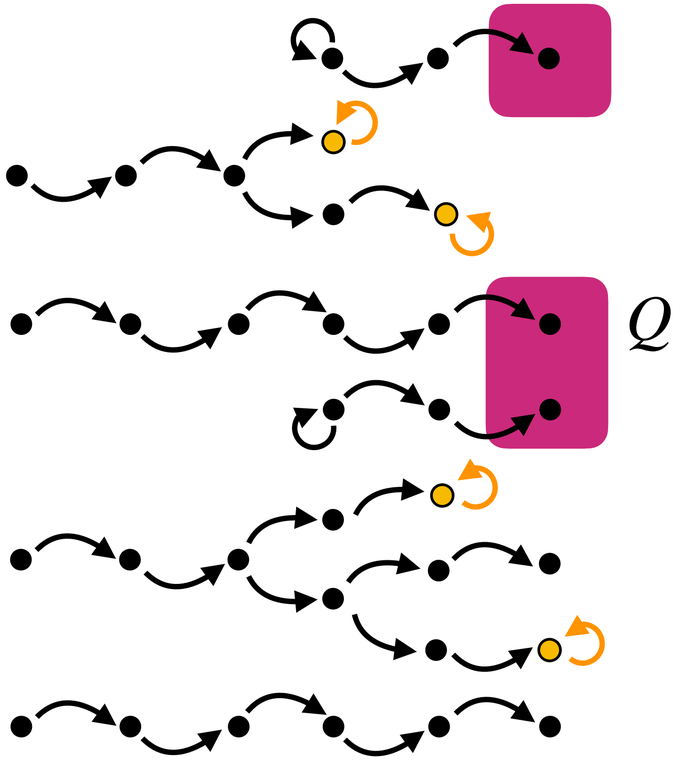
States outside Q
are reachable

By duality: (un)reachability



$(\texttt{false})^{\infty}\ r\ \lfloor Q \rfloor$

every state in Q
is unreachable

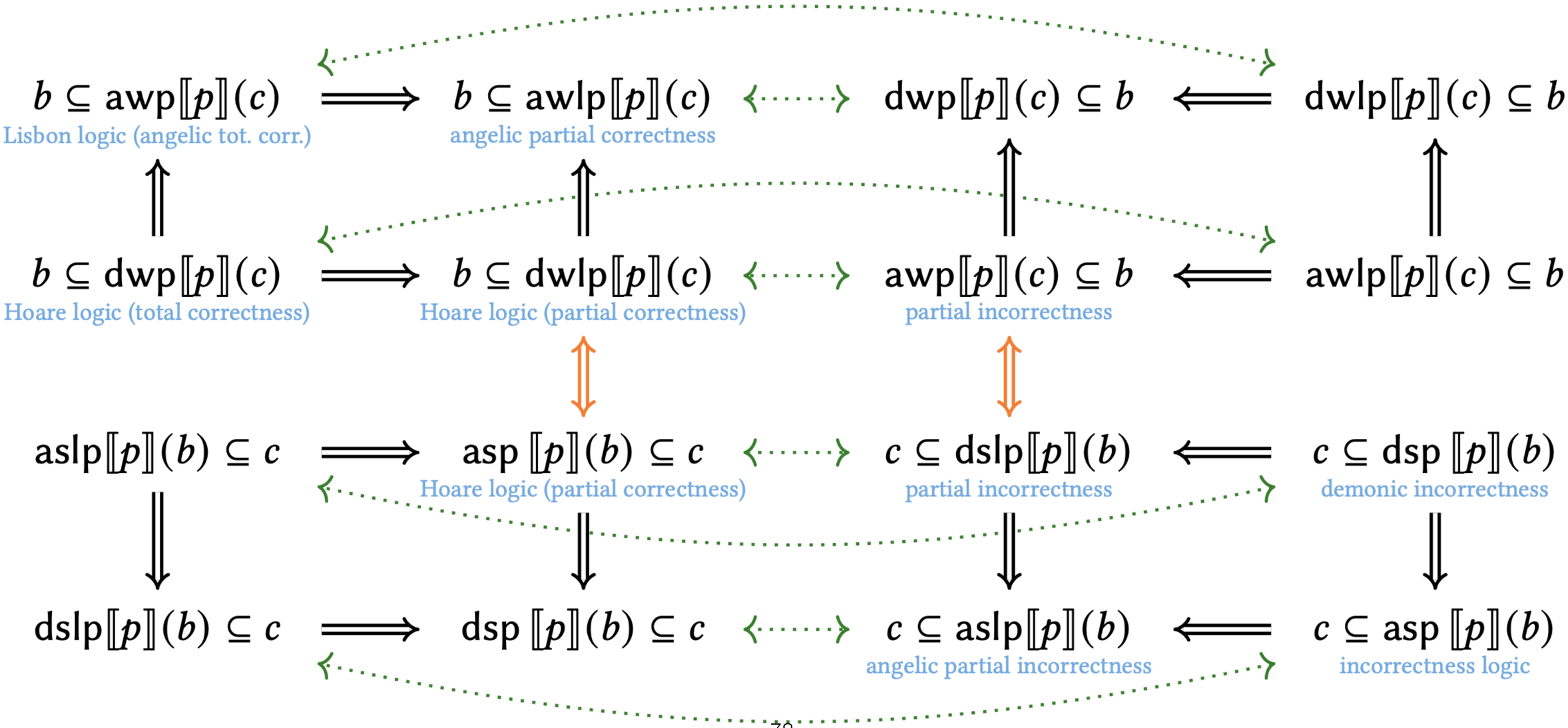


$(\texttt{false})^{\infty}\ r\ \lceil Q \rceil$

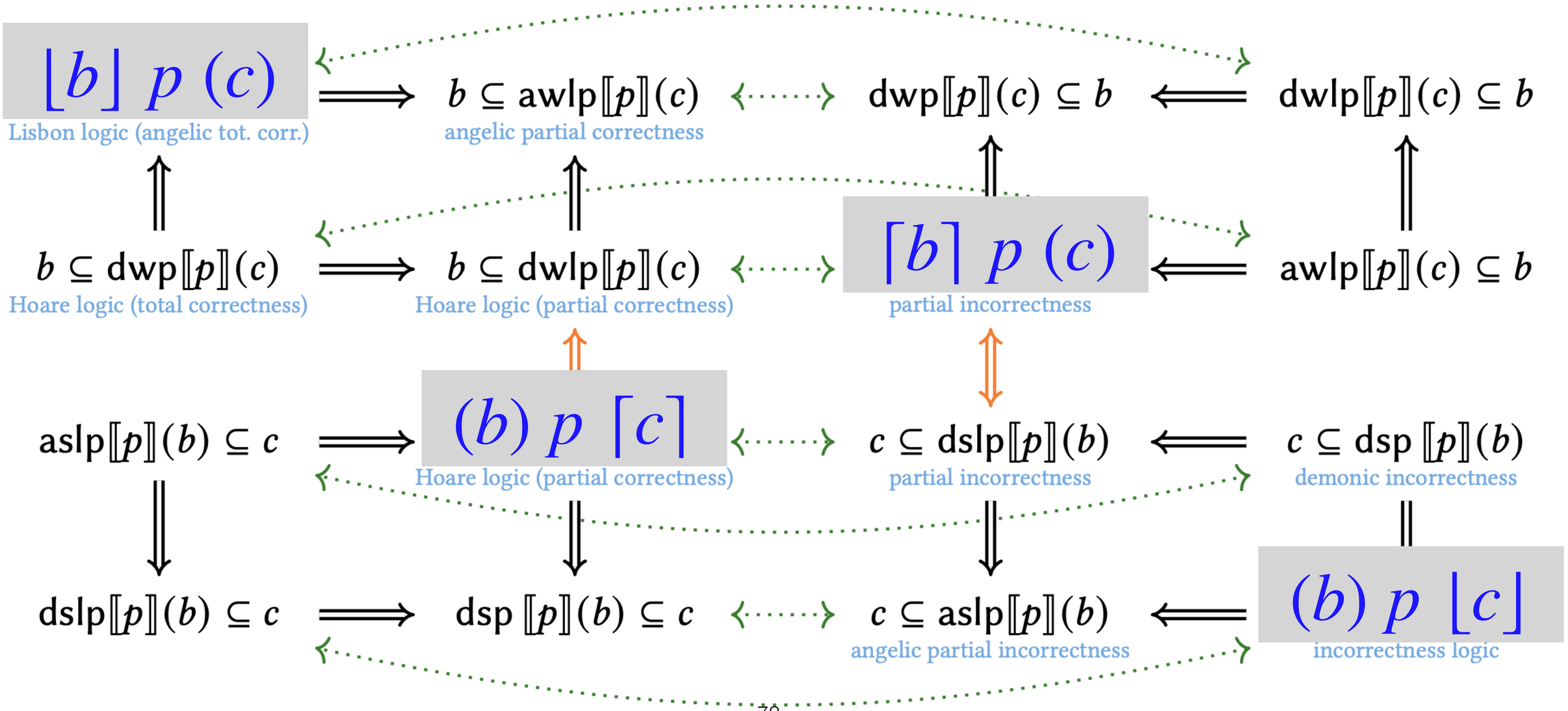
States outside Q
are reachable

Related Work and Conclusions

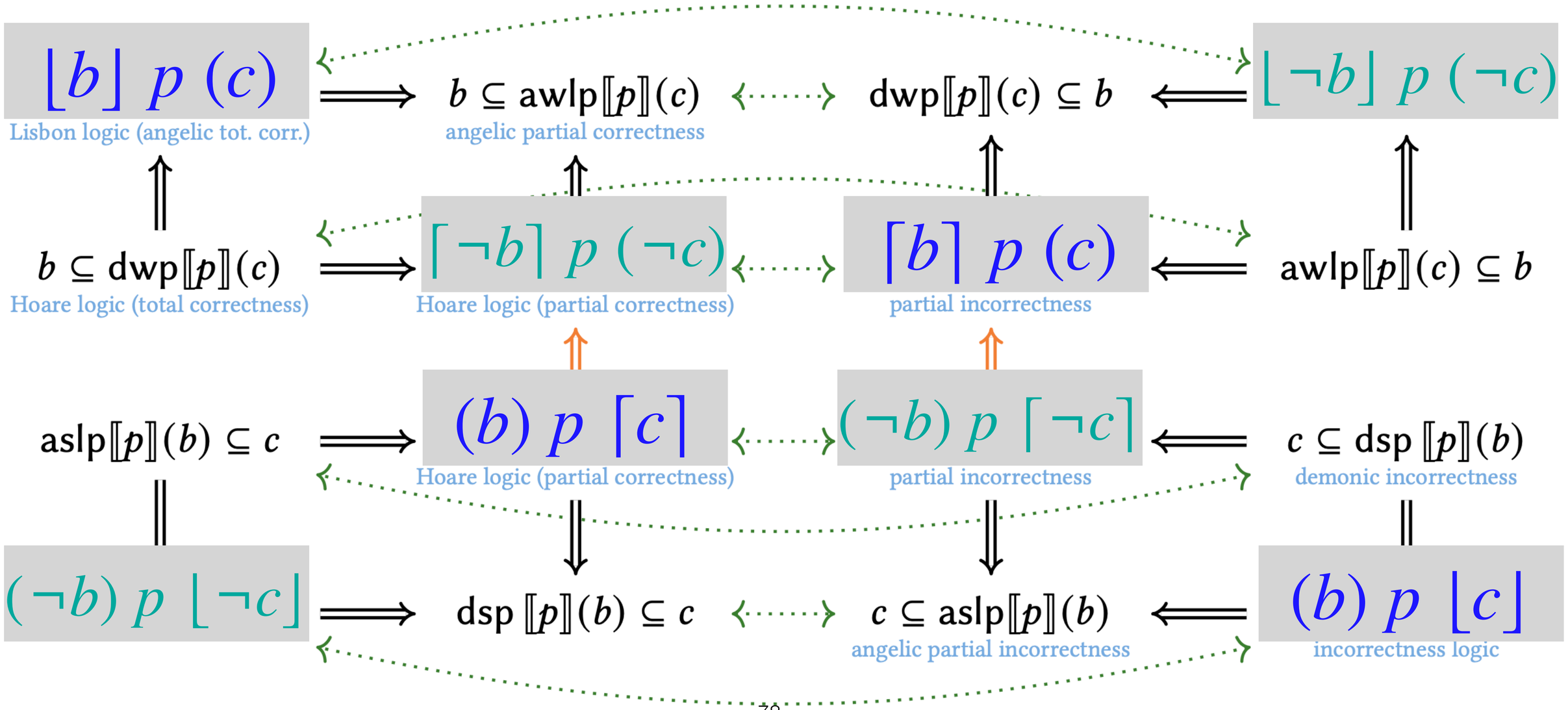
[POPL25] Verscht and Kaminski



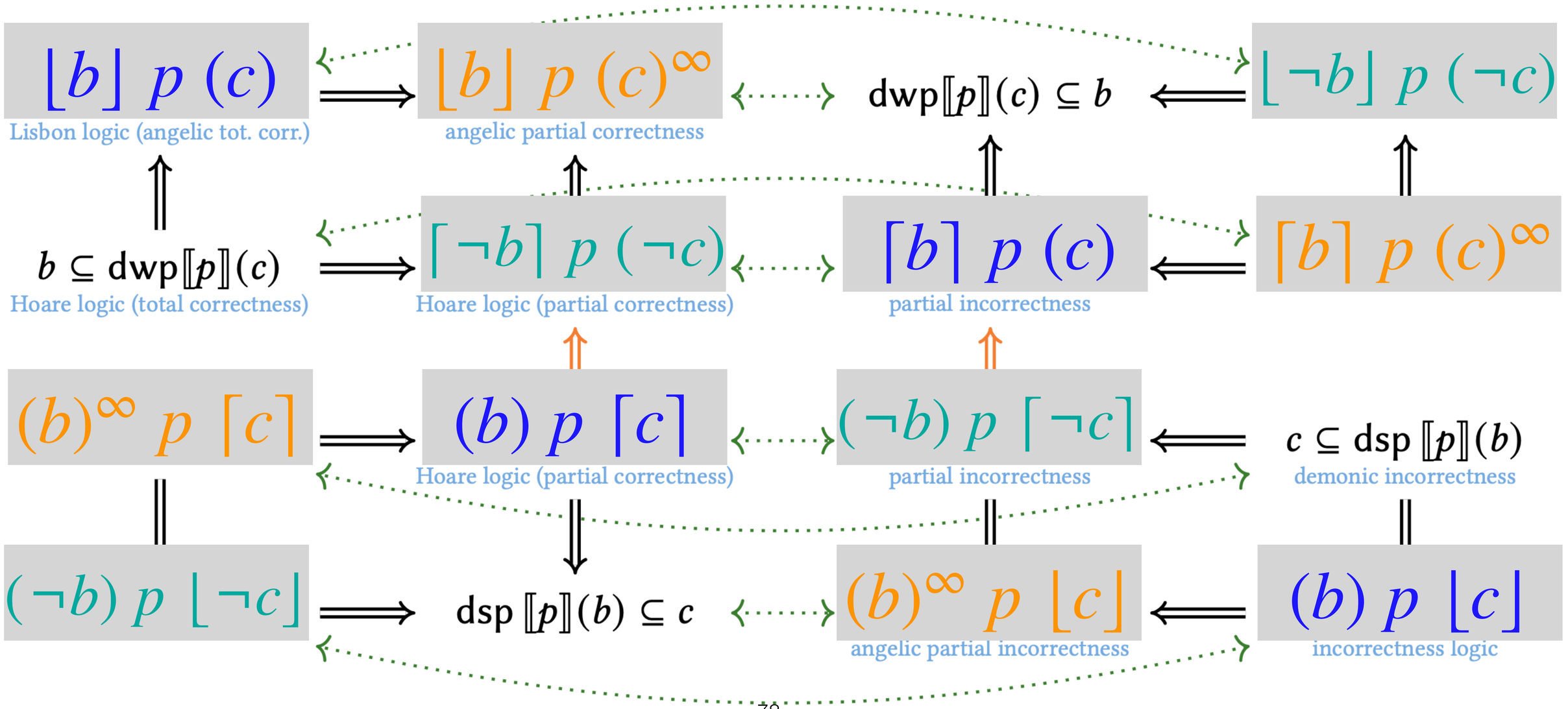
[POPL25] Verscht and Kaminski



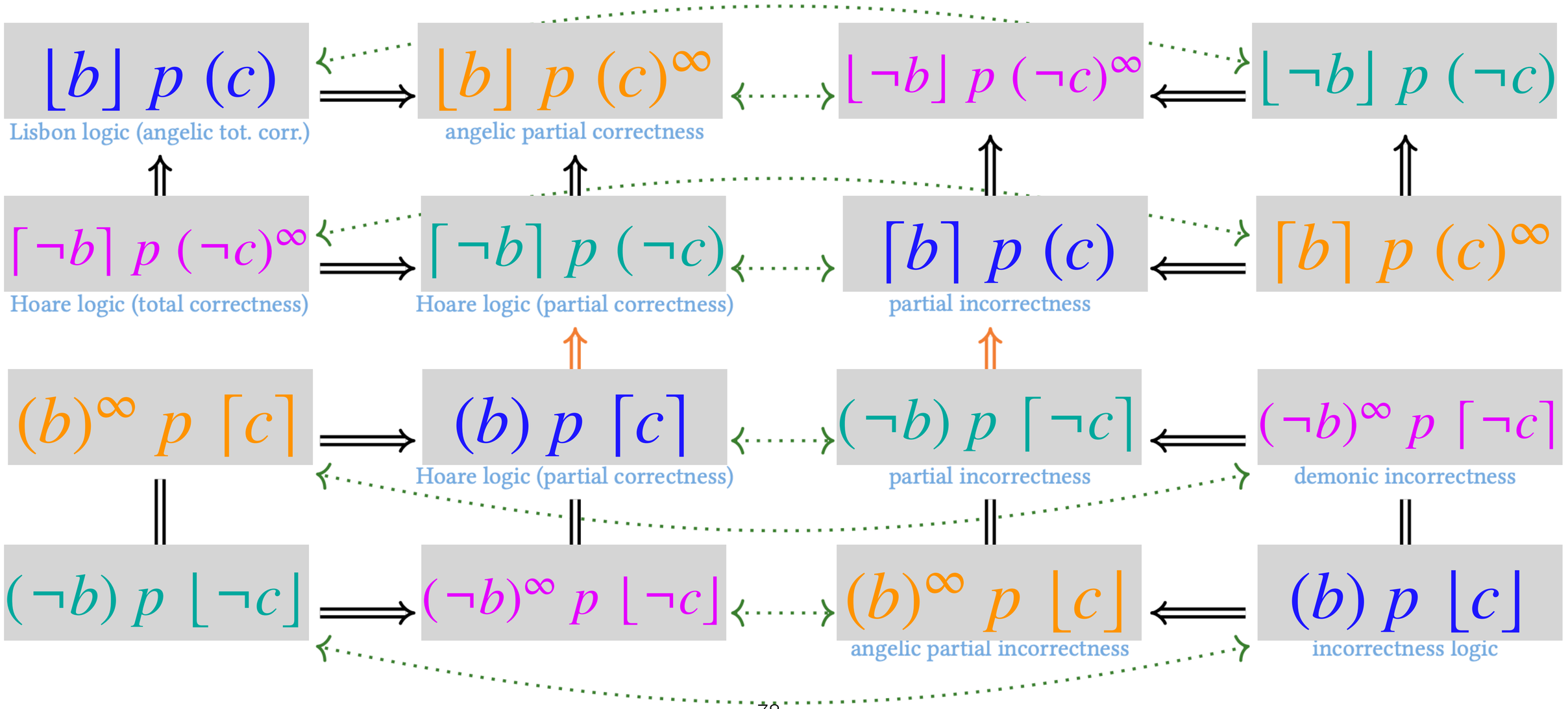
[POPL25] Verscht and Kaminski



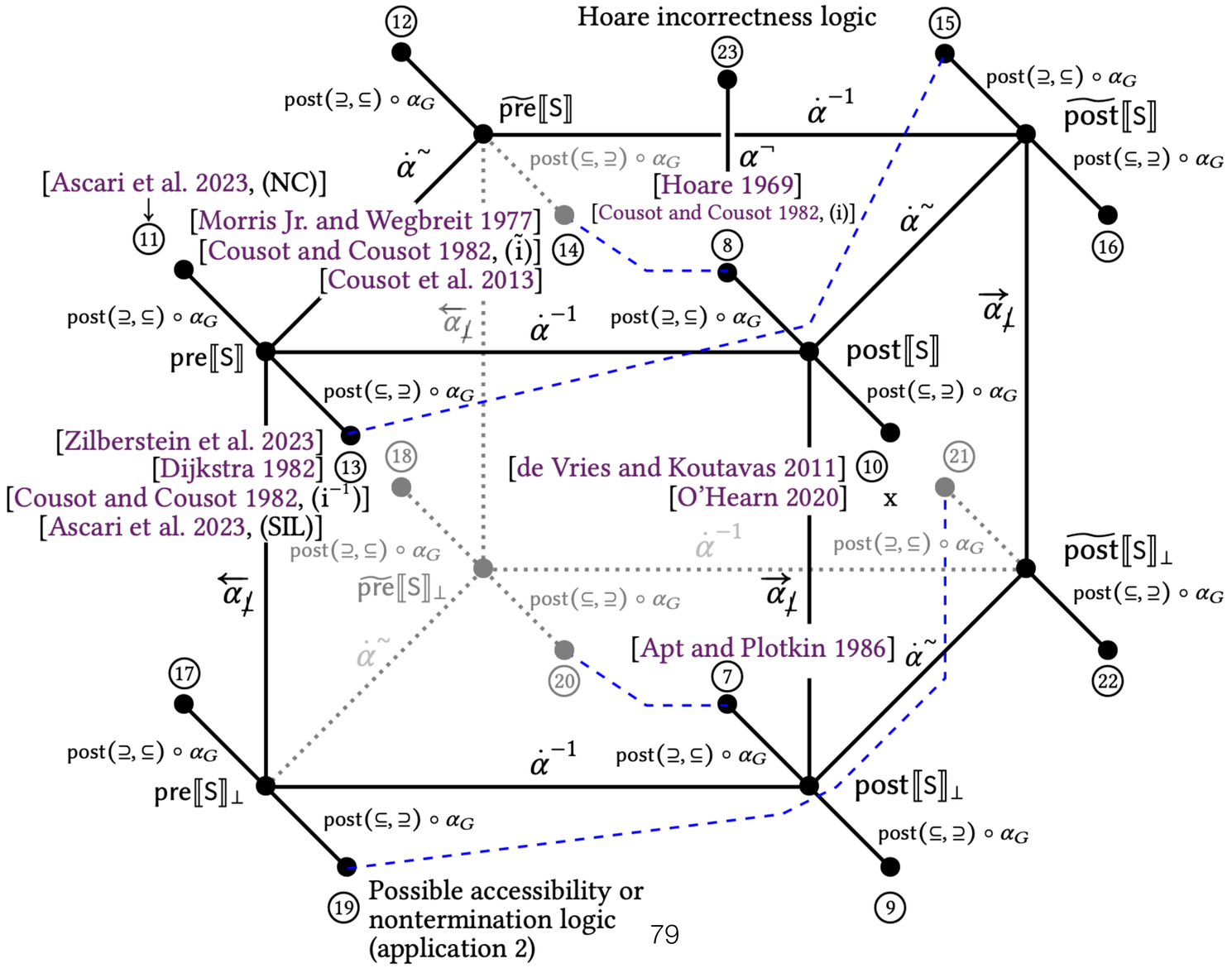
[POPL25] Verscht and Kaminski



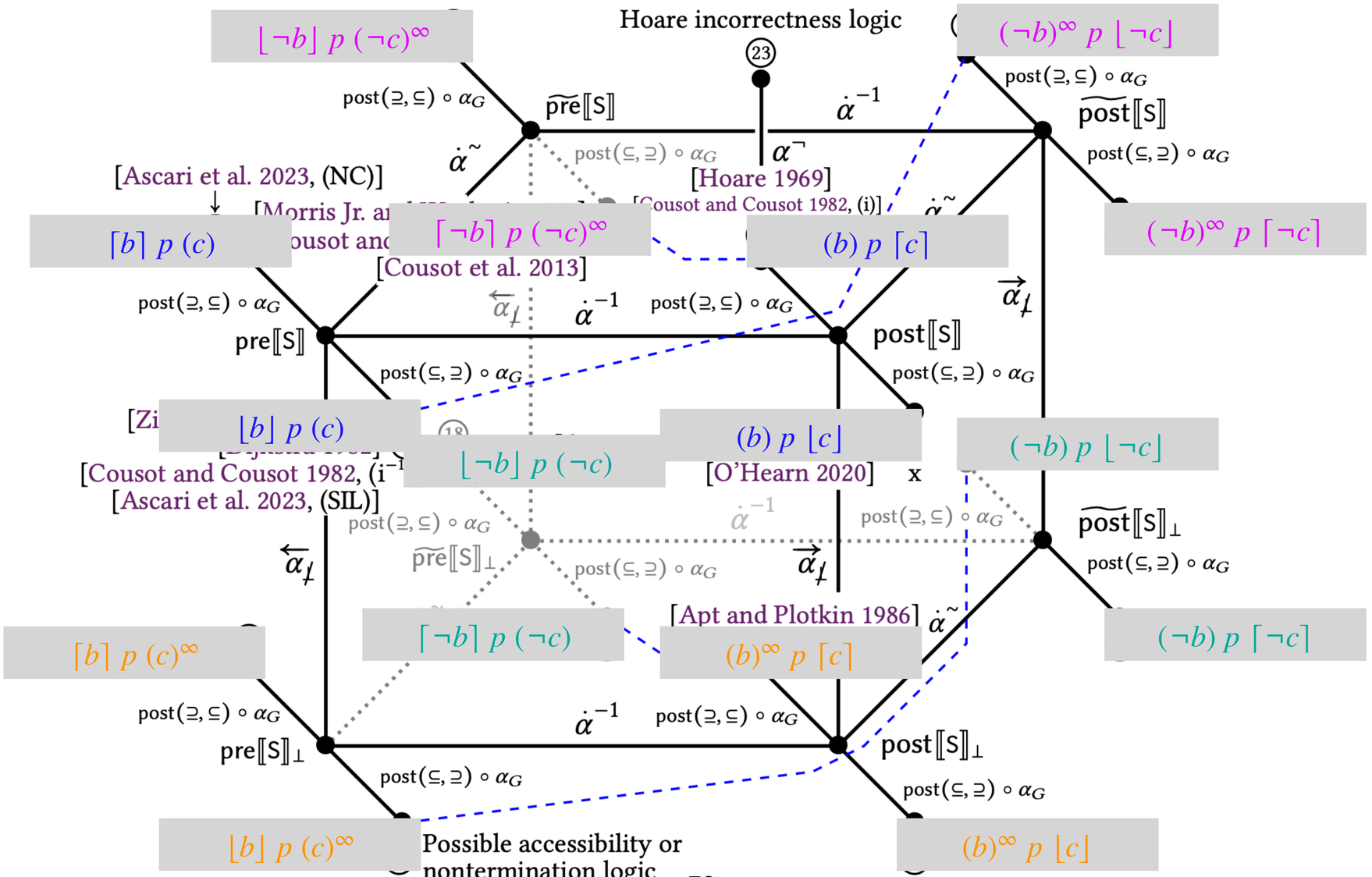
[POPL25] Verscht and Kaminski



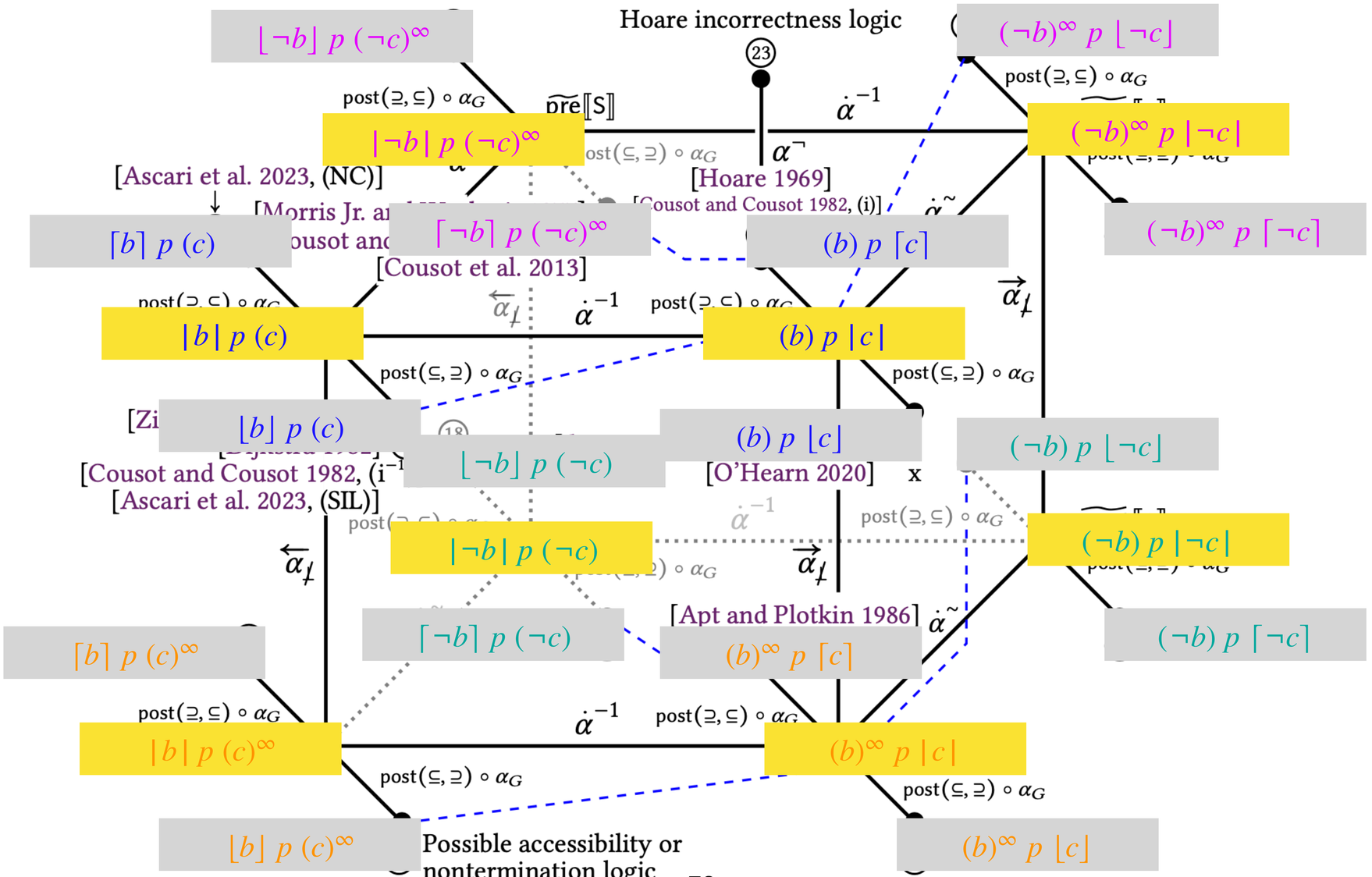
[POPL24] Cousot



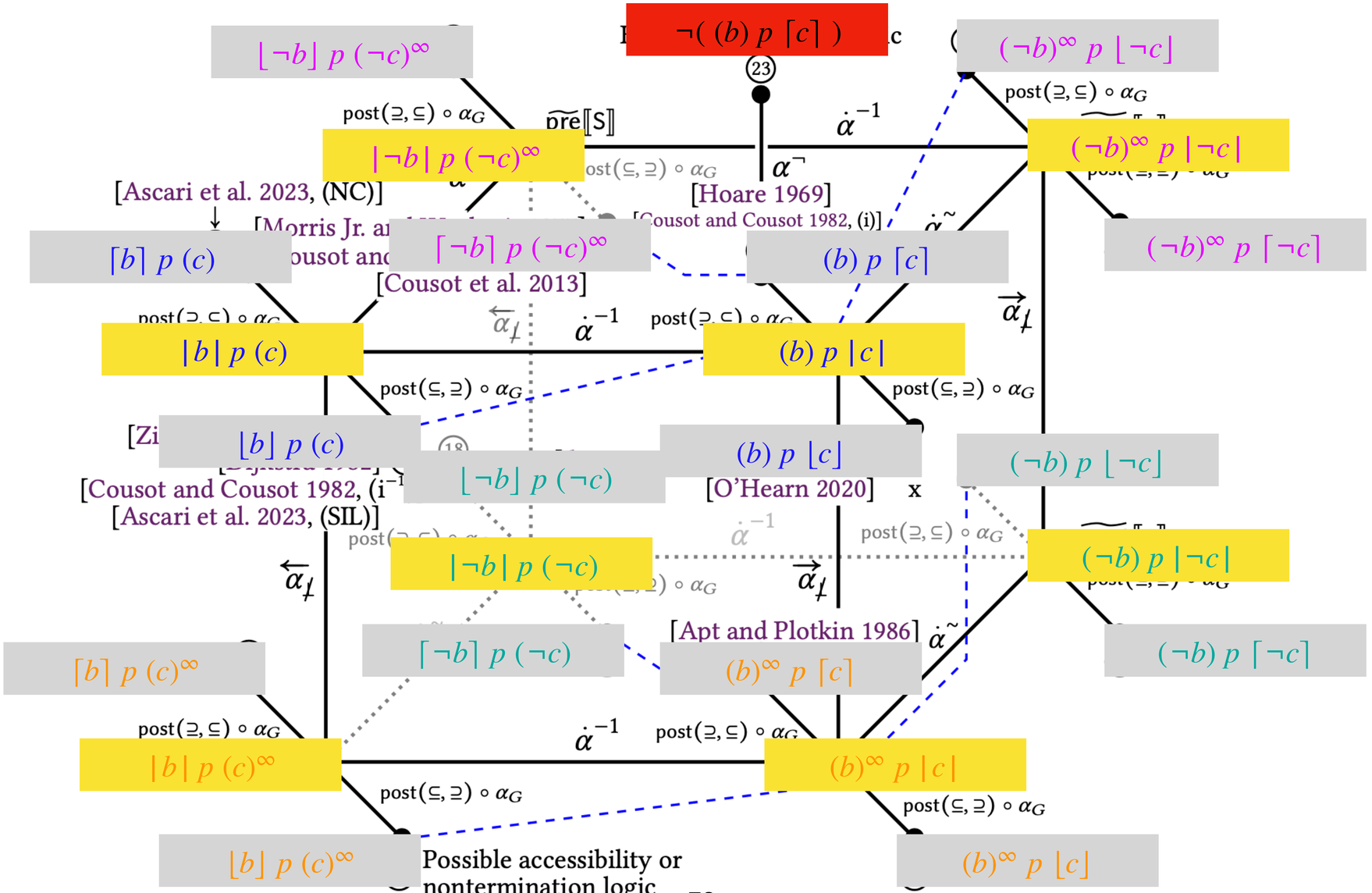
[POPL24] Cousot



[POPL24] Cousot

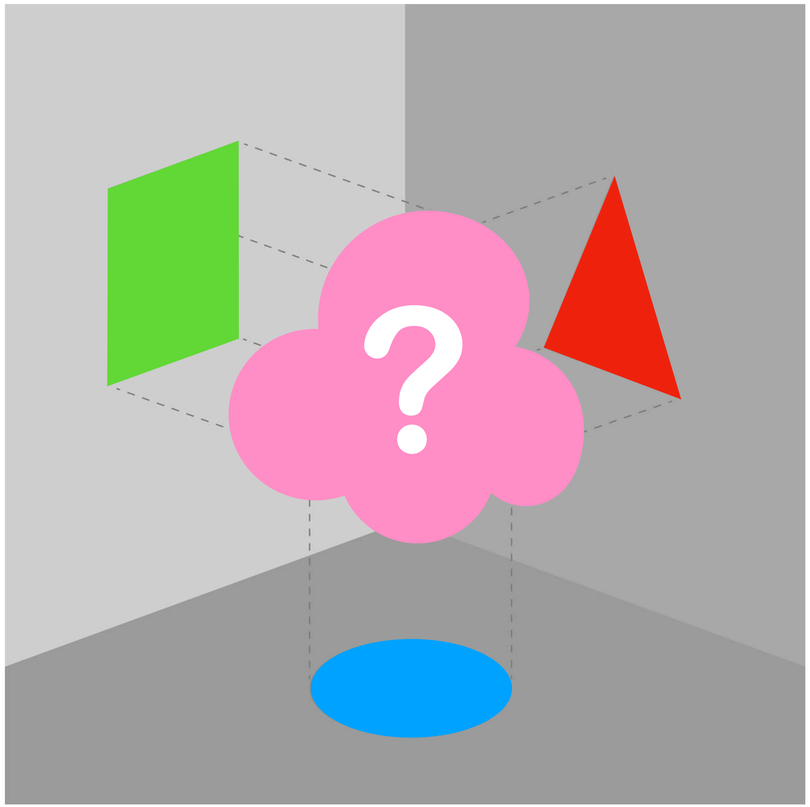


[POPL24] Cousot



Conclusion

No single view reveals the whole picture



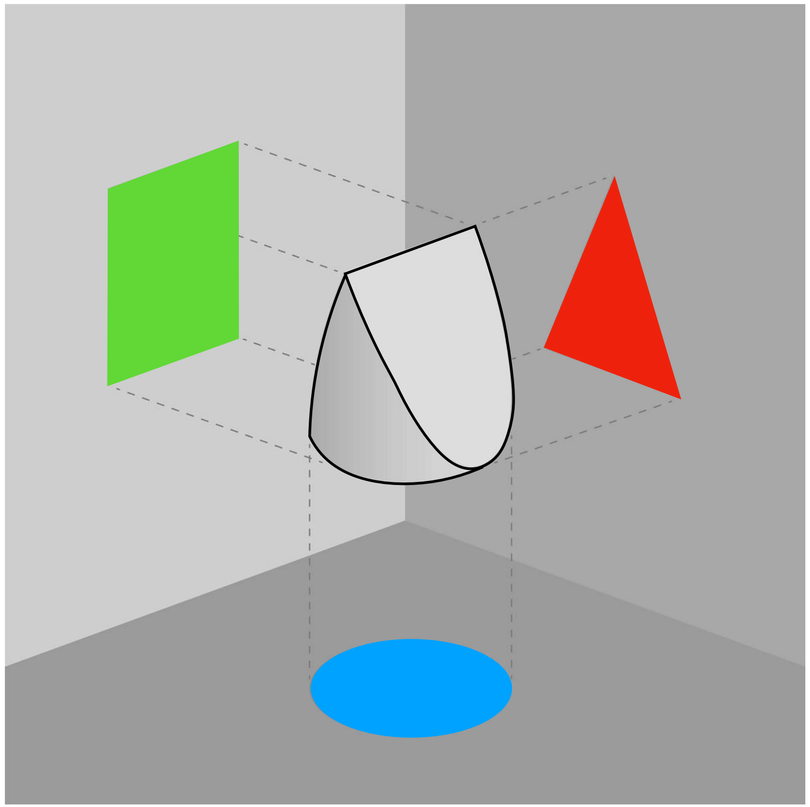
- Novel compact, standard notation for code summaries
- A single proof system for all program logics
- Also accounts for exact triples (predicate transformers)
- Also accounts for combined triples

$\lceil P \rceil\ r\ \lceil Q \rceil$ $\lfloor P \rfloor\ r\ \lceil Q \rceil$ $\lceil P \rceil\ r\ \lfloor Q \rfloor$ $\lfloor P \rfloor\ r\ \lfloor Q \rfloor$

- Reduced set of rules by exploiting similarities
- Also accounts for may-termination and reachability

Conclusion

No single view reveals the whole picture



- Novel compact, standard notation for code summaries
- A single proof system for all program logics
- Also accounts for exact triples (predicate transformers)
- Also accounts for combined triples

$\lceil P \rceil\ r\ \lceil Q \rceil$ $\lfloor P \rfloor\ r\ \lceil Q \rceil$ $\lceil P \rceil\ r\ \lfloor Q \rfloor$ $\lfloor P \rfloor\ r\ \lfloor Q \rfloor$

- Reduced set of rules by exploiting similarities
- Also accounts for may-termination and reachability

Ongoing works

$\lfloor P \rfloor\ r\ \lfloor Q \rfloor$ $\lceil P \rceil\ r\ \lceil Q \rceil$ $\lfloor P \rfloor\ r\ \lceil Q \rceil$ $\lceil P \rceil\ r\ \lfloor Q \rfloor$

- Encompass convex triples (under + over) and abstract interpretation

$$(P)\ r\ \lfloor Q_1 \,|\, Q_2 \rceil$$

$$\lfloor p \,|\, P^{\#} \rceil\ r\ \lfloor q \,|\, Q^{\#} \rceil$$

- Extend our approach to:
  - Separation logics and their toolsets
  - Concurrent programs
  - Probabilistic programs (in the style of Outcome Logic)
  - Relational logic

What ChatGPT got from this talk...

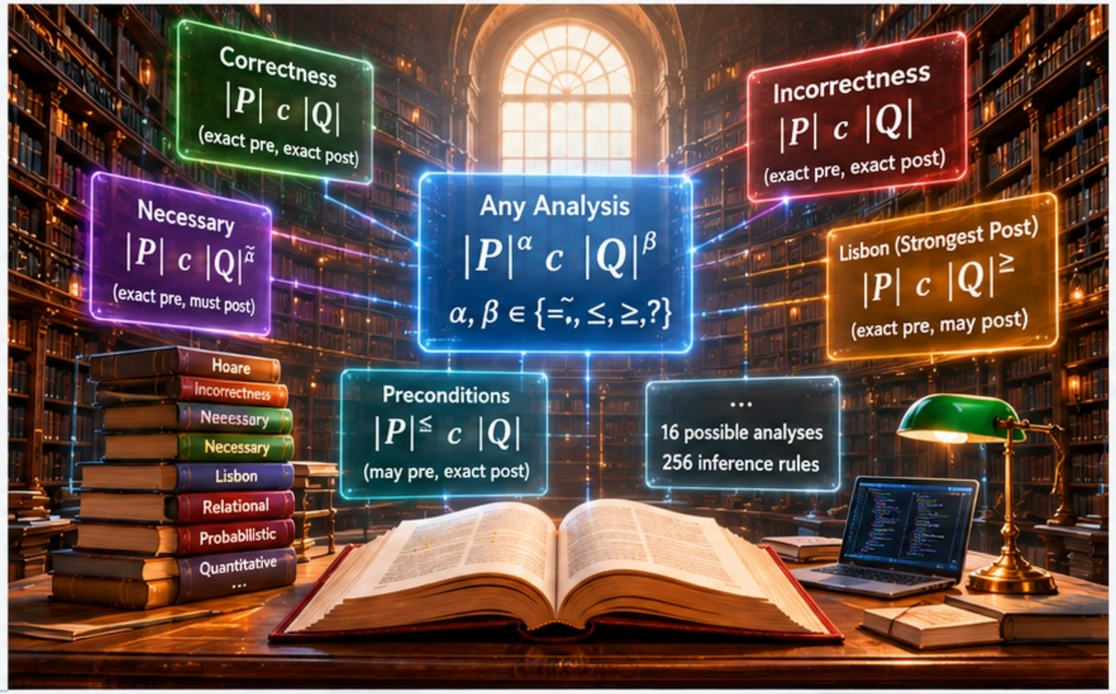
A Library that Knows Everything

One place. All the information. All the reasons.

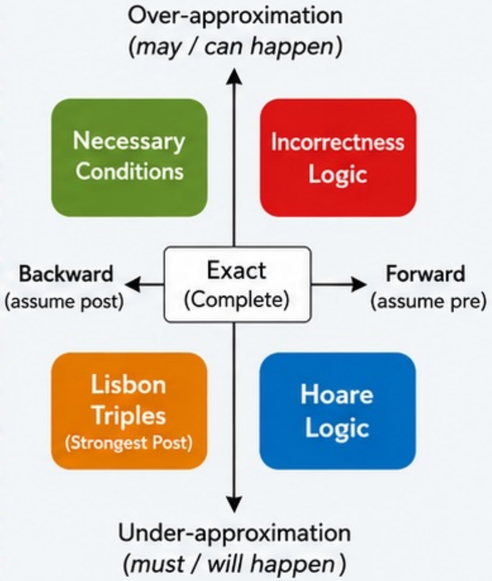
The Universal Library of Program Analyses

For every program, every input, every behavior.

- **Correctness** — When it always does the right thing
- **Incorrectness** — When it can go wrong
- **Preconditions** — When it must never be called
- **Necessary Conditions** — What must be true in the past
- **Lisbon Triples (Strongest Post)** — What it can possibly produce
- And all the combinations in between



A Unified Semantic Space



Example: a simple function

```
int safe_div(int a, :int b) {
  if (b == 0) {
     error("division by zero");
  }
  return a / b;
}
```

Correctness

$|b \neq 0|$ safe_div(a,b) $|r = a/b|$

If $b \neq 0$ it always returns a/b

Incorrectness

$|b = 0|$ safe_div(a,b) [error("division by zero")]

If $b = 0$ it can raise an error

Two summaries, two complementary views of the same program.

One function. Many truths. Many reasons.

Our logic collects all of them in a single, uniform notation $|P|^{\alpha}\ c\ |Q|^{\beta}$ so you can use the right one for your proof, your analysis, your need.

Verify your code | Prevent your bugs | Document your APIs | Understand every behavior

Take home message

Different approaches are often seen as competing theories, but they can offer complementary views of the same system:
cooperation is MUCH better than conflict

Take home message

Different approaches are often seen as competing theories, but they can offer complementary views of the same system:
cooperation is MUCH better than conflict

less dogmatism,
be open minded!

⌊Thanks⌋ for your ⌈attention!⌉

Mentioned references:

- [Hoa69] An Axiomatic Basis for Computer Programming. Commun. ACM 12, 10 (1969). C. A. R. Hoare.
- [VMCAI13] Automatic Inference of Necessary Preconditions. P. Cousot, R. Cousot, M. Fähndrich, F. Logozzo.
- [POPL20] Incorrectness logic. P. O'Hearn.
- [POPL24] Calculational design of [in] correctness transformational program logics by abstract interpretation. P. Cousot.
- [OOPSLA24] Non-termination Proving at Scale. A. Raad, J. Vanegue, P. O'Hearn.
- [OOPSLA25] Revealing Sources of (Memory) Errors via Backward Analysis. F. Ascari, R. Bruni, R. Gori, F. Logozzo.
- [POPL25] A Taxonomy of Hoare-Like Logics. L. Verscht, B. L. Kaminski.
- [POPL26] U-Turn: Enhancing Incorrectness Analysis by Reversing Direction. F. Ascari, R. Bruni, R. Gori, A. Raad.