More intensional versions of Rice's Theorem

Jean-Yves Moyen[1] Jakob Grue Simonsen[1]
Jean-Yves.Moyen@lipn.univ-paris13.fr

[1]Datalogisk Institut
University of Copenhagen

Supported by the Marie Curie action "Walgo" program H2020-MSCA-IF-2014, number 655222 and the Danish Council for Independent Research *Sapere Aude* grant "Complexity via Logic and Algebra" (COLA).

April 2-3 2016

Rice's and Asperti-Rice's Theorems

Rice's Theorem

A cornerstone of computability.

Theorem (Rice, '53)

Any non-trivial and extensional set of programs is undecidable.

Rice's Theorem

A cornerstone of computability.

Theorem (Rice, '53)

Any non-trivial and extensional set of programs is undecidable.

extensional: do not separate programs computing the same function: $\mathtt{p} \in \mathcal{P}, \mathtt{q} \notin \mathcal{P} \Rightarrow [\![\mathtt{p}]\!] \neq [\![\mathtt{q}]\!]$.

Rice's Theorem

A cornerstone of computability.

Theorem (Rice, '53)

Any non-trivial and extensional set of programs is undecidable.

extensional: do not separate programs computing the same function: $\mathtt{p} \in \mathcal{P}$, $\mathtt{q} \notin \mathcal{P} \Rightarrow [\![\mathtt{p}]\!] \neq [\![\mathtt{q}]\!]$.

Proof.

$\mathtt{p} \neq$ infinite loop, $\mathtt{p} \in \mathcal{P}$, loop $\notin \mathcal{P}$.
$\mathtt{q}'(x) = \mathtt{q}(0); \mathtt{p}(x)$.
$\mathtt{q}' \in P \Leftrightarrow \mathtt{q}(0)$ terminates. □

The power of Rice

Rice's Theorem allows to prove undecidability of a wide range of sets of programs:

- programs which (don't) terminate on input 0;
- programs which return 42 on input 54;
- programs which return an even result on any prime input;
- programs computing a total function;
- programs computing a bijection;
- ...

The power of Rice

Rice's Theorem allows to prove undecidability of a wide range of sets of programs:

- programs which (don't) terminate on input 0;
- programs which return 42 on input 54;
- programs which return an even result on any prime input;
- programs computing a total function;
- programs computing a bijection;
- ...

But it cannot be used for *intensional* sets that depend on **program** behaviour (complexity, ...)

Extensional equivalence

"Extensionality" of sets defines an equivalence on programs, the extensional equivalence (or Rice's equivalence):
$\mathtt{p}\mathfrak{R}\mathtt{q} \Leftrightarrow [\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$.

Extensional equivalence

"Extensionality" of sets defines an equivalence on programs, the extensional equivalence (or Rice's equivalence):
$\mathtt{p}\mathfrak{R}\mathtt{q} \Leftrightarrow [\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$.
Rice's Theorem now state that:

- $\mathfrak{R}$ is undecidable;
- any equivalence less precise than $\mathfrak{R}$ is undecidable.

Extensional equivalence

"Extensionality" of sets defines an equivalence on programs, the extensional equivalence (or Rice's equivalence):
$\mathtt{p}\mathfrak{R}\mathtt{q} \Leftrightarrow [\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$.
Rice's Theorem now state that:

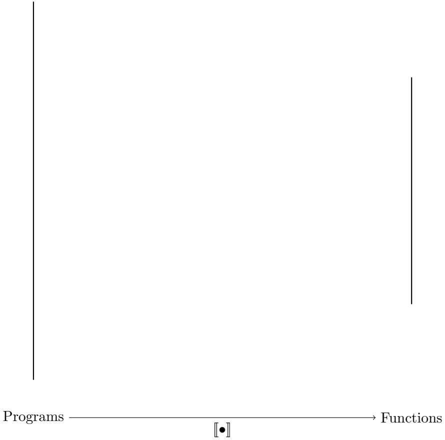
- $\mathfrak{R}$ is undecidable;
- any equivalence less precise than $\mathfrak{R}$ is undecidable.

Theorem (Rice, again)

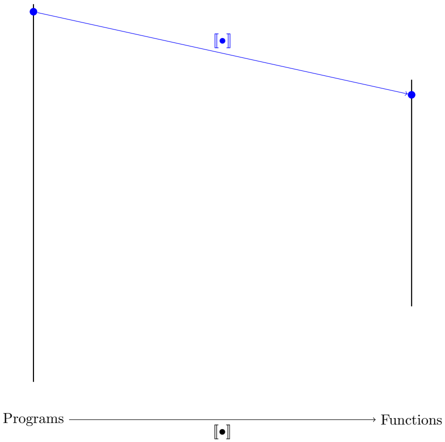
Any non-trivial set of programs which is the union of classes of $\mathfrak{R}$ is undecidable.

What about equivalences more precise than $\mathfrak{R}$?

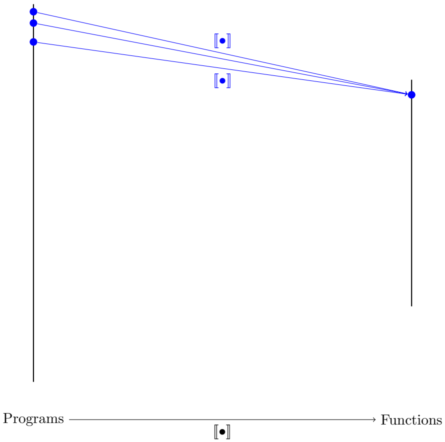
The semantics tunnel (1)



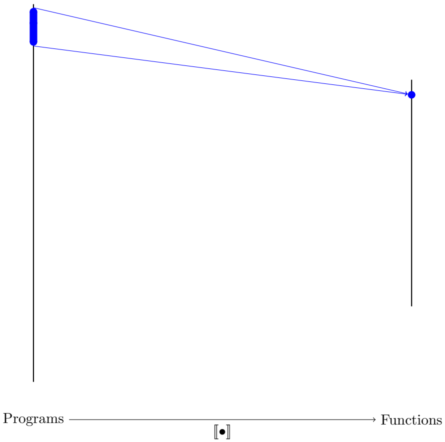
The semantics tunnel (1)



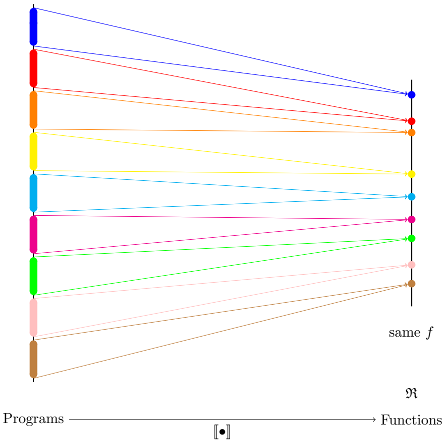
The semantics tunnel (1)



The semantics tunnel (1)



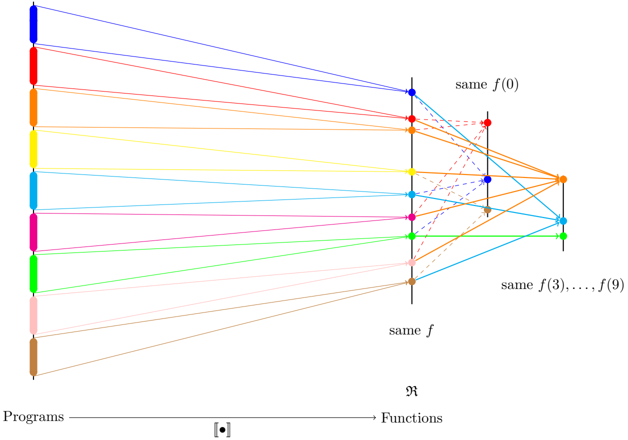
The semantics tunnel (1)



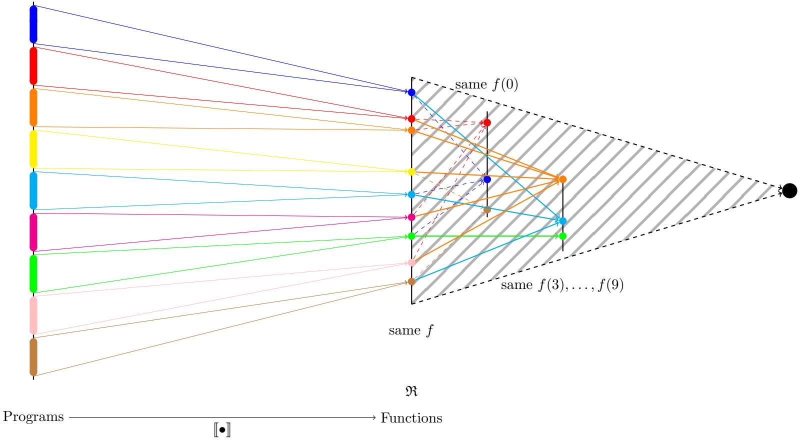
The semantics tunnel (1)

tensionaln

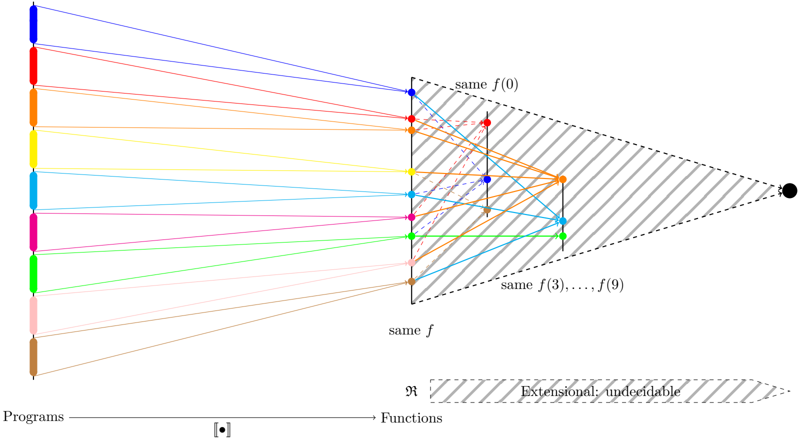
The semantics tunnel (1)



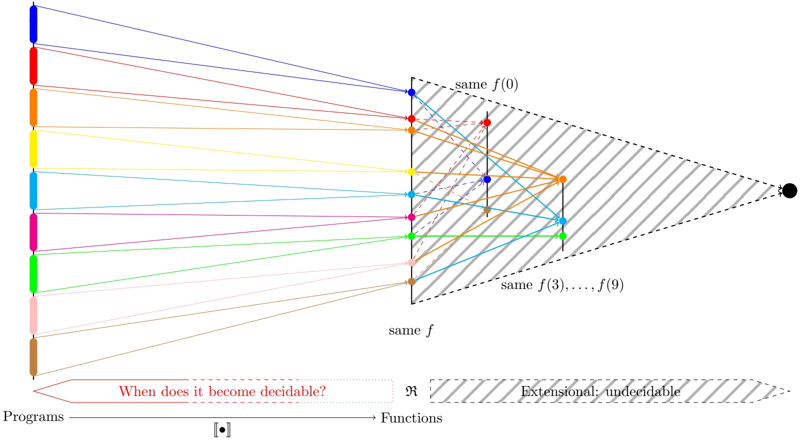
The semantics tunnel (1)



The semantics tunnel (1)



The semantics tunnel (1)



Asperti-Rice's Theorem

A first intensional version of Rice's Theorem.

$\mathtt{p}\mathfrak{A}\mathtt{q} \Leftrightarrow [\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$ and $\mathtt{cplx}(\mathtt{p}) = \Theta(\mathtt{cplx}(\mathtt{q}))$ ("clique")

Asperti-Rice's Theorem

A first intensional version of Rice's Theorem.

$\mathtt{p}\mathfrak{A}\mathtt{q} \Leftrightarrow [\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$ and $\mathtt{cplx}(\mathtt{p}) = \Theta(\mathtt{cplx}(\mathtt{q}))$ ("clique")

Theorem (Asperti, '08)

Any non-trivial set of programs which is the union of classes of $\mathfrak{A}$ is undecidable.

Asperti-Rice's Theorem

A first intensional version of Rice's Theorem.
$\mathtt{p}\mathfrak{A}\mathtt{q} \Leftrightarrow [\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$ and $\mathtt{cplx}(\mathtt{p}) = \Theta(\mathtt{cplx}(\mathtt{q}))$ ("clique")

Theorem (Asperti, '08)

Any non-trivial set of programs which is the union of classes of $\mathfrak{A}$ is undecidable.

The set of programs computing the sorting function in polynomial time.

Asperti-Rice's Theorem

A first intensional version of Rice's Theorem.

$\mathtt{p}\mathfrak{A}\mathtt{q} \Leftrightarrow [\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$ and $\mathtt{cplx}(\mathtt{p}) = \Theta(\mathtt{cplx}(\mathtt{q}))$ ("clique")

Theorem (Asperti, '08)

Any non-trivial set of programs which is the union of classes of $\mathfrak{A}$ is undecidable.

The set of programs computing the sorting function in polynomial time.

Proof: Same as Rice!

$\mathtt{p}$ not equivalent to infinite loop. $\mathtt{q}'(x) = \mathtt{q}(0); \mathtt{p}(x)$.

Asperti-Rice's Theorem

A first intensional version of Rice's Theorem.

$\mathtt{p}\mathfrak{A}\mathtt{q} \Leftrightarrow [\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$ and $\mathtt{cplx}(\mathtt{p}) = \Theta(\mathtt{cplx}(\mathtt{q}))$ ("clique")

Theorem (Asperti, '08)

Any non-trivial set of programs which is the union of classes of $\mathfrak{A}$ is undecidable.

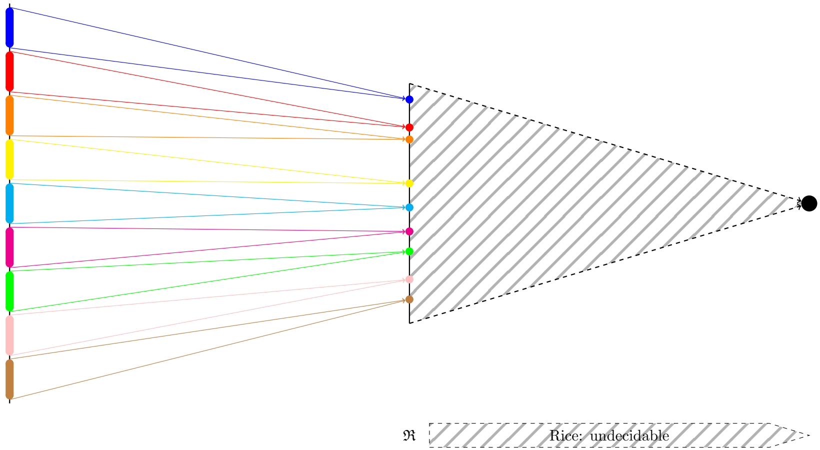
The set of programs computing the sorting function in polynomial time.

Proof: Same as Rice!

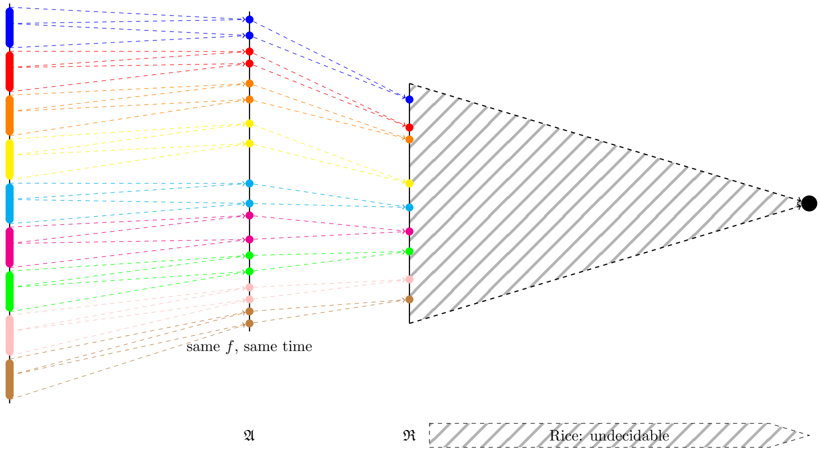
$\mathtt{p}$ not equivalent to infinite loop. $\mathtt{q}'(x) = \mathtt{q}(0); \mathtt{p}(x)$.

If $\mathtt{q}(0)$ terminates, it does so with a **fixed** complexity so $\mathtt{p}$ and $\mathtt{q}'$ have the same complexity up to an additive factor. □

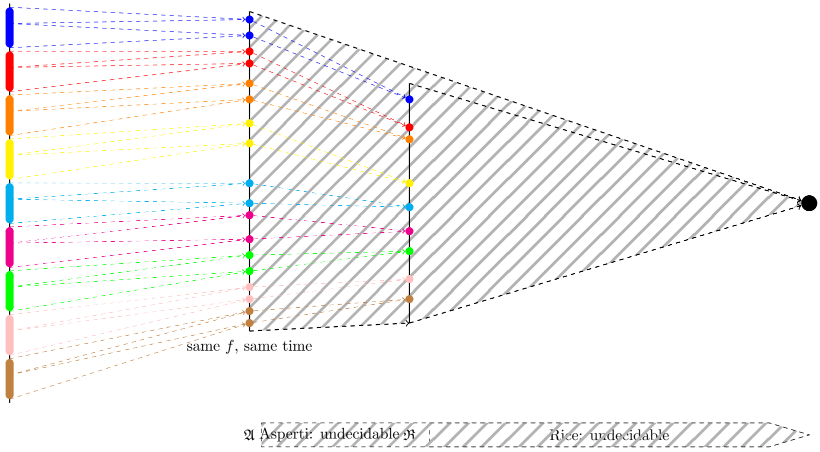
The semantics tunnel (2)



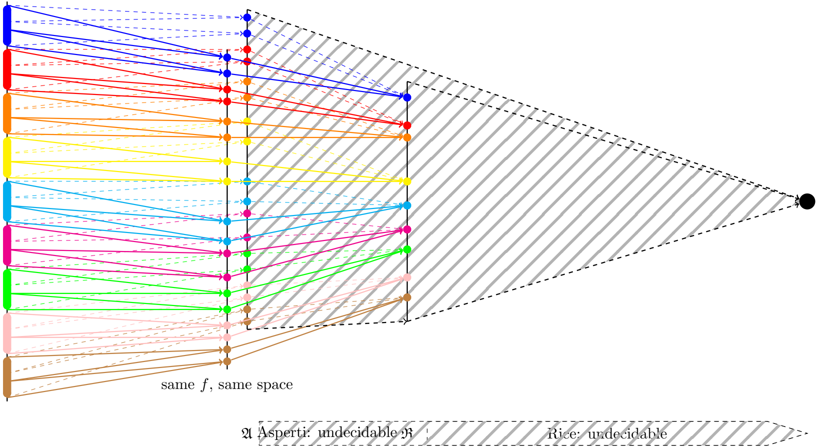
The semantics tunnel (2)



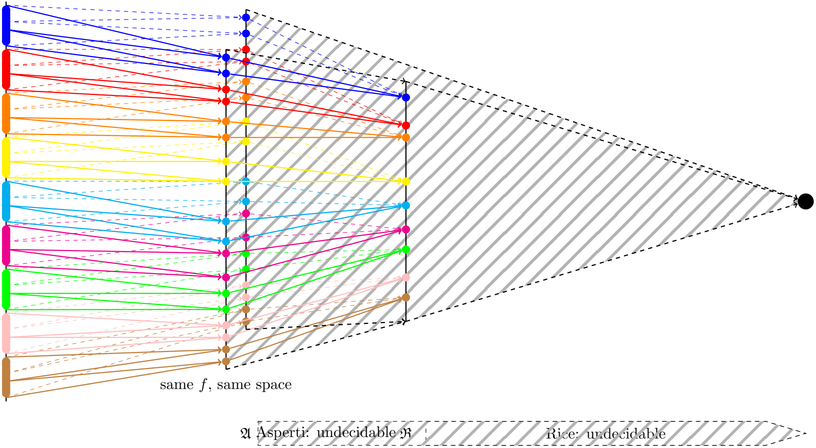
The semantics tunnel (2)



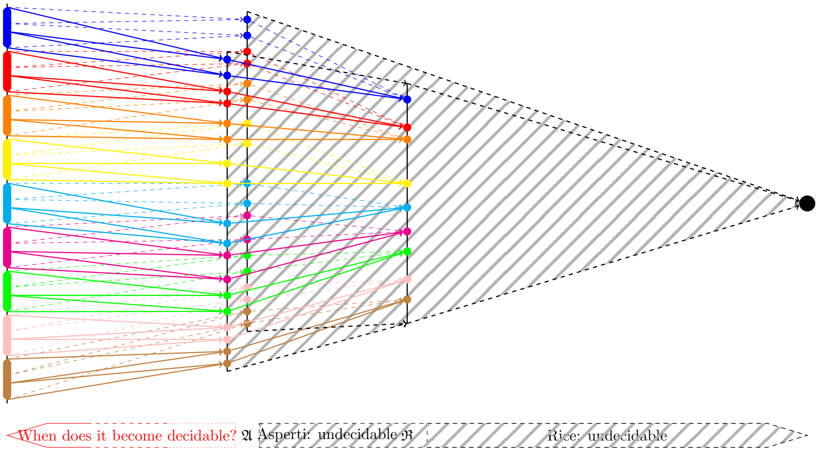
The semantics tunnel (2)



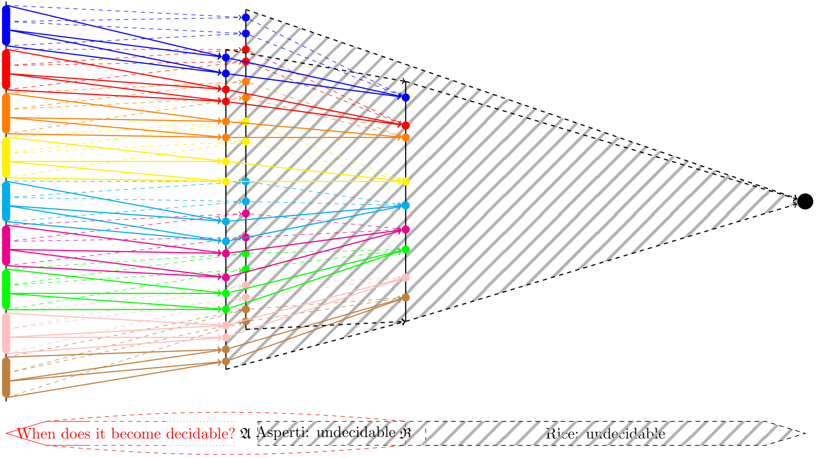
The semantics tunnel (2)



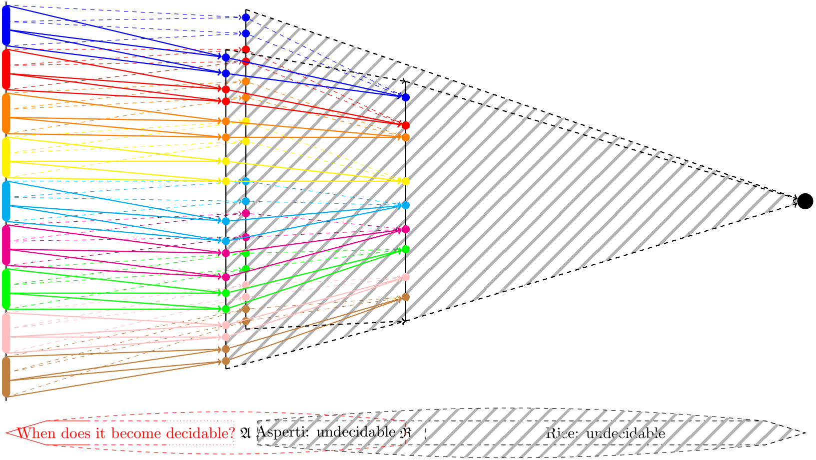
The semantics tunnel (2)



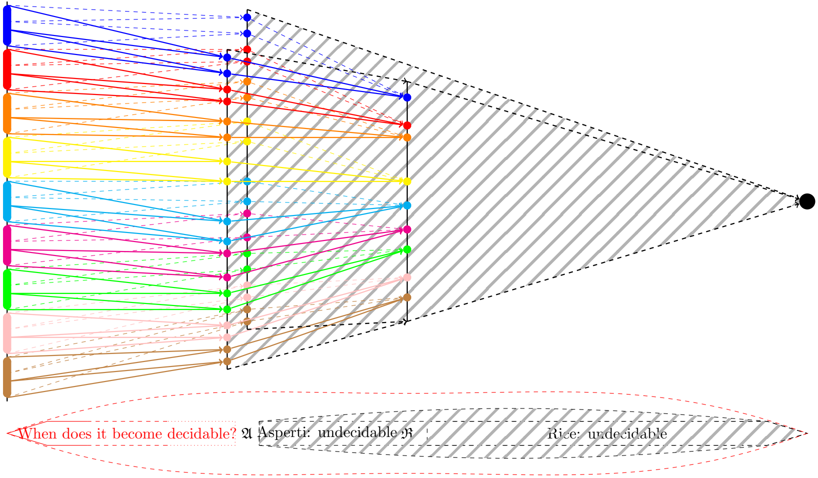
The semantics tunnel (2)



The semantics tunnel (2)

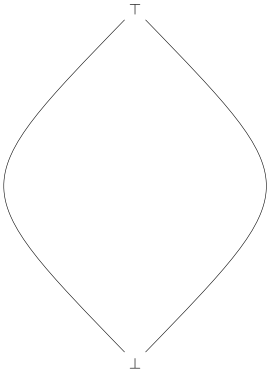


The semantics tunnel (2)



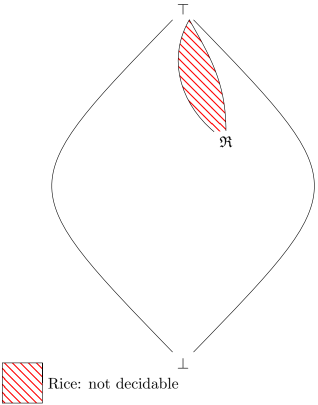
The equivalences lattice
Not the subject of today's talk!

- The set of all equivalences is a complete lattice.
- $\bot$: equality, $\top$: one class with everything.



The equivalences lattice
Not the subject of today's talk!

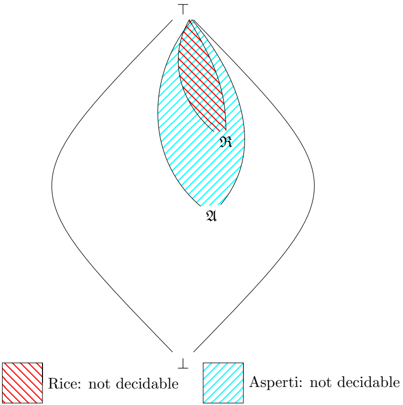
- The set of all equivalences is a complete lattice.
- $\bot$: equality, $\top$: one class with everything.
- Rice: nothing in the principal filter at $\mathfrak{R}$ is decidable.



The equivalences lattice

Not the subject of today's talk!

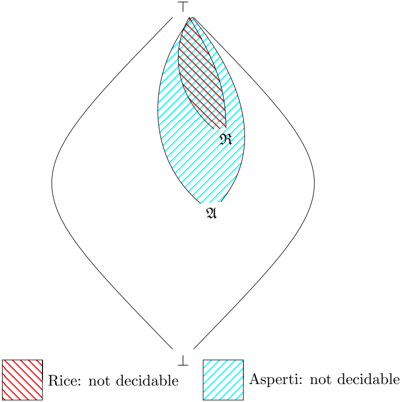
- The set of all equivalences is a complete lattice.
- $\bot$: equality, $\top$: one class with everything.
- Rice: nothing in the principal filter at $\mathfrak{R}$ is decidable.
- Asperti: nothing in the principal filter at $\mathfrak{A}$ is decidable.



The equivalences lattice

Not the subject of today's talk!

- The set of all equivalences is a complete lattice.
- $\bot$: equality, $\top$: one class with everything.
- Rice: nothing in the principal filter at $\mathfrak{R}$ is decidable.
- Asperti: nothing in the principal filter at $\mathfrak{A}$ is decidable.



Complicated and interesting structure.
Ongoing works with J. G. Simonsen and J. Avery.

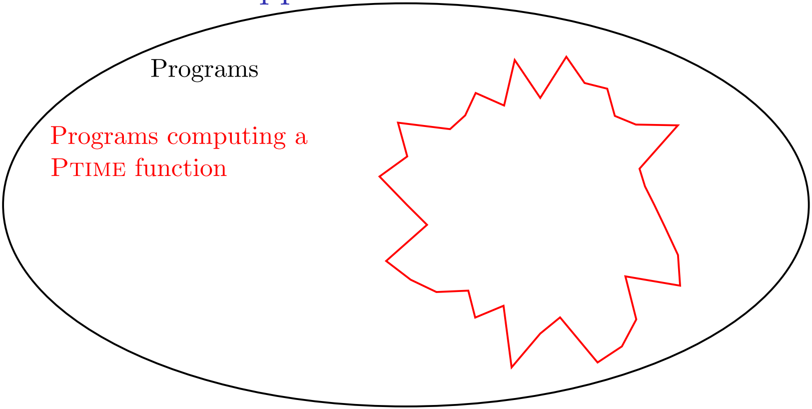
First generalisation

Today's talk

Two generalisations of Rice's Theorem relaxing the extensionality condition.

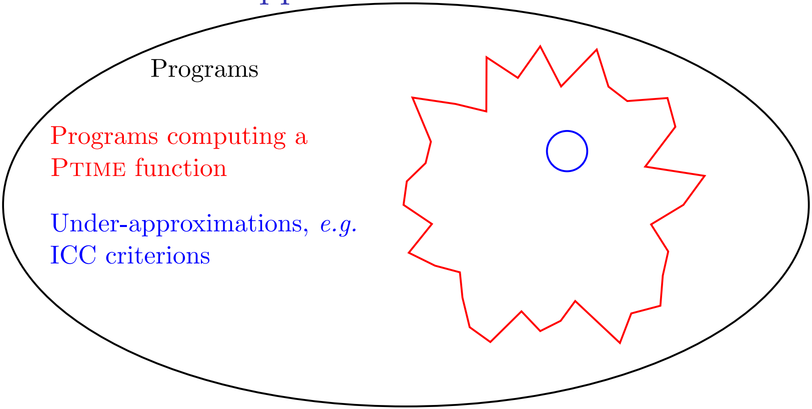
1. Rather than searching equivalences more precises than $\mathfrak{R}$, keep it but consider sets that are not just union of classes.
2. Try the same approach with a wide range of others equivalences.

Under- and over- approximations



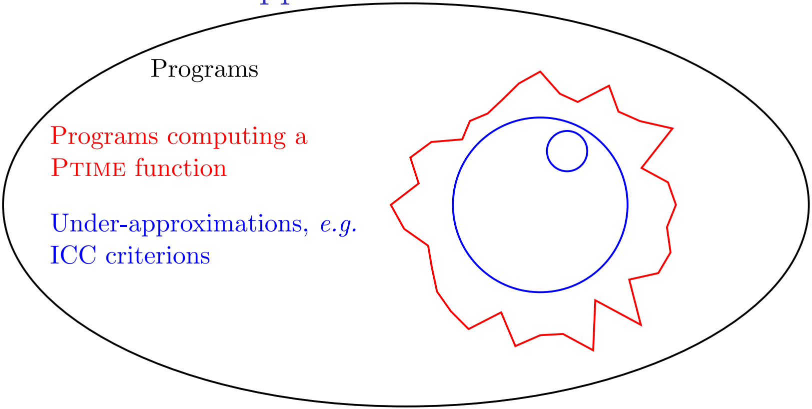
 is **not** PPTIME, the set of polytime **programs**. It is undecidable by Rice's Theorem.

Under- and over- approximations



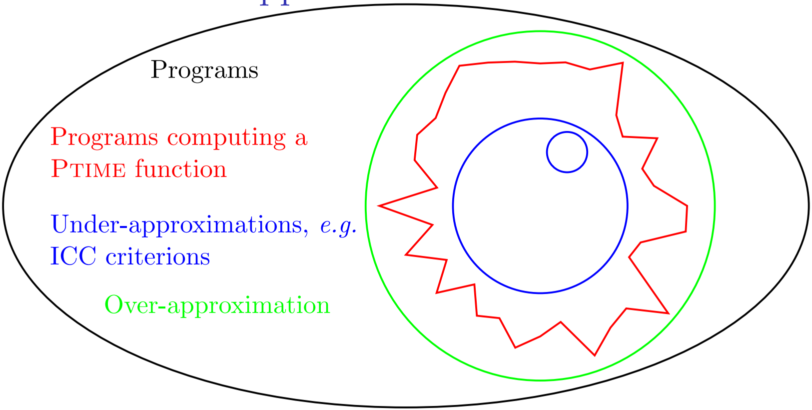
○ is an ICC criterion if it captures one program for each PTIME function.

Under- and over- approximations



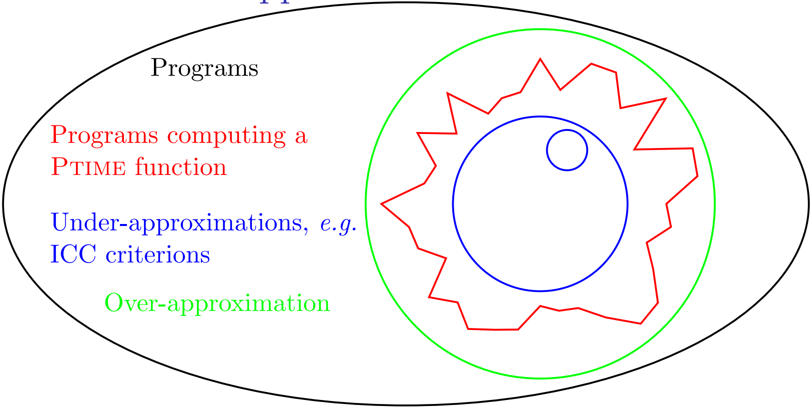
○ is an ICC criterion if it captures one program for each Ptime function.

Under- and over- approximations



Can ○ be decidable and “small enough”?

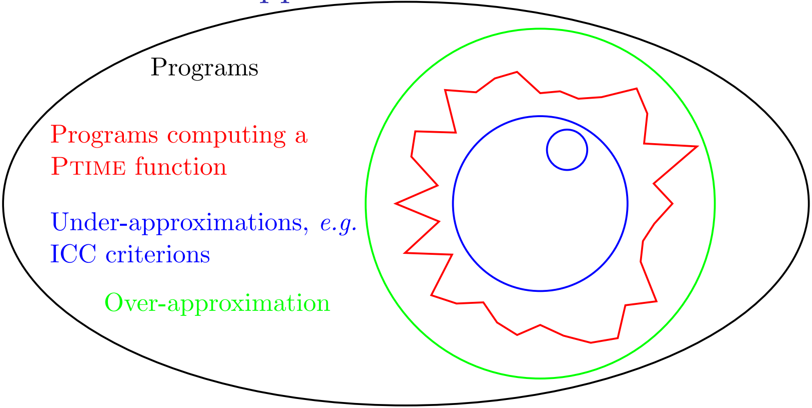
Under- and over- approximations



Can ○ be decidable and "small enough"?

Upper bound: $\mathtt{p} \in \bigcirc \Rightarrow [\![\mathtt{p}]\!] \in$ PTIME.

Under- and over- approximations



Can $\bigcirc$ be decidable and "small enough"?

Upper bound: $\texttt{p} \in \bigcirc \Rightarrow [\![\texttt{p}]\!] \in$ PTIME.

Lower bound: $\texttt{p} \notin \bigcirc \Rightarrow [\![\texttt{p}]\!] \notin$ PTIME.

Vocabulary

A set of programs is:

- *non-trivial* if it is neither empty, nor the set of all programs.
- *extensional* if it is the union of classes of $\mathfrak{R}$;
- *partially extensional* (for F) if it contains all the programs with $[\![\mathsf{p}]\!] \in F$ (over approximation).
- *extensionally complete* (for F) if it contains one program for each $f \in F$.
- *extensionally sound* (for F) if it contains only programs with $[\![\mathsf{p}]\!] \in F$ (under approximation).
- an *ICC characterisation* (of F) if it is both extensionally sound and complete for F.
- *extensionally universal* if it is extensionally complete for the set of computable partial functions.

First Result

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

First Result

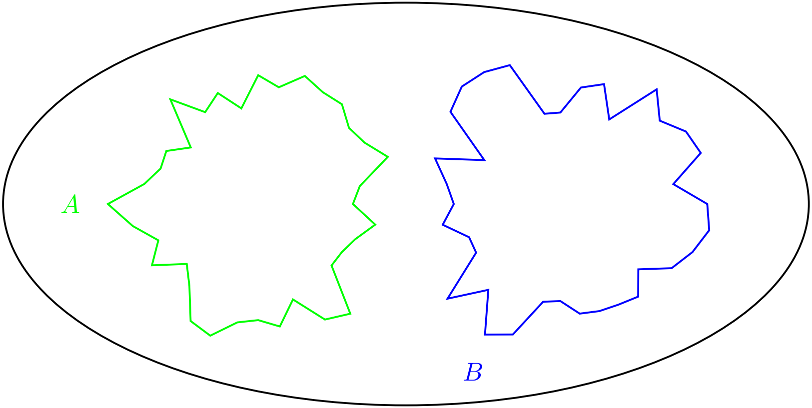
Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

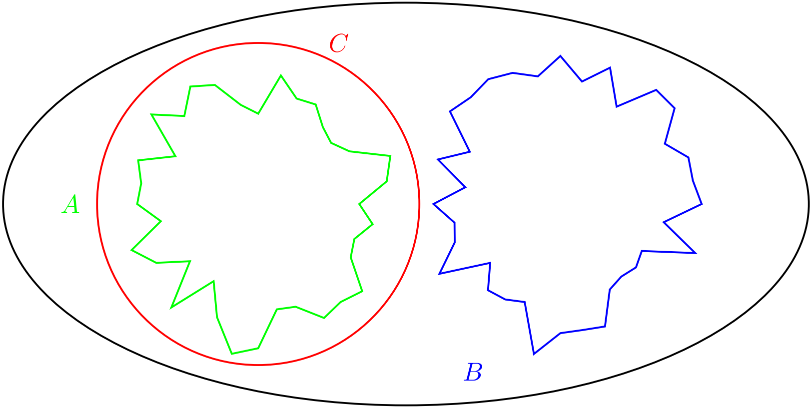
Definition

Two sets A and B are *recursively separable* if there exists C decidable with $A \subset C$ and $B \cap C = \emptyset$.

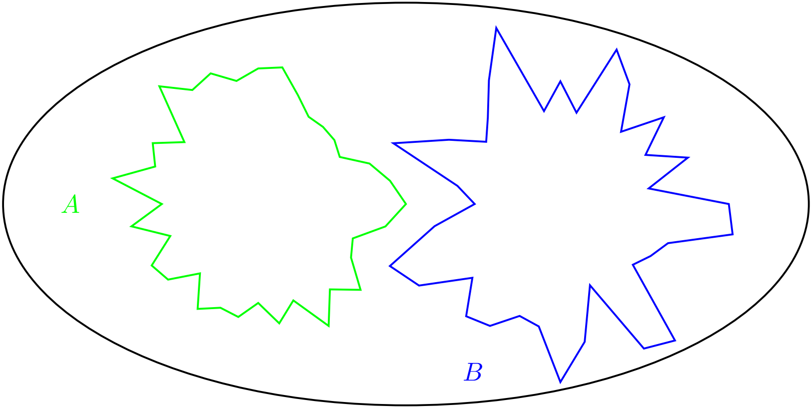
First Result



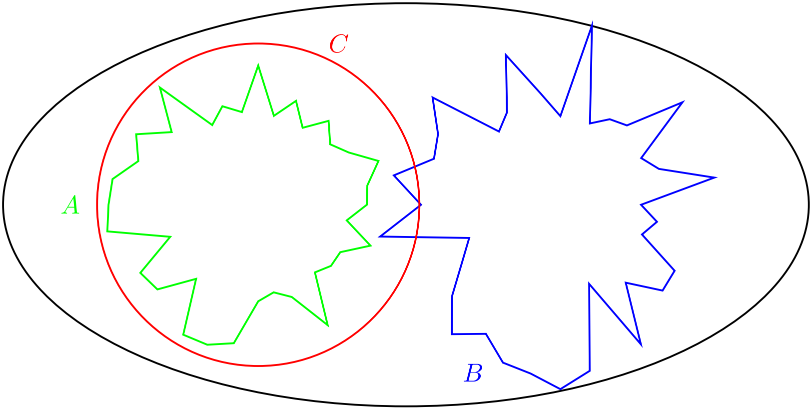
First Result



First Result



First Result



First Result

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Definition

Two sets A and B are *recursively separable* if there exists C decidable with $A \subset C$ and $B \cap C = \emptyset$.

First Result

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Definition

Two sets A and B are *recursively separable* if there exists C decidable with $A \subset C$ and $B \cap C = \emptyset$.

"Decidable over-approximation of A that does not intersect B."

First Result

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Definition

Two sets A and B are *recursively separable* if there exists C decidable with $A \subset C$ and $B \cap C = \emptyset$.

"Decidable over-approximation of A that does not intersect B."

Example

$A = \{\, \mathtt{p} \;:\; [\![\mathtt{p}]\!]\,(0) = 0 \,\}$
$B = \{\, \mathtt{p} \;:\; [\![\mathtt{p}]\!]\,(0) \notin \{0, \bot\} \,\}$

recursively inseparable

First Result

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Example

$$A = \{\, \mathtt{p} \;:\; [\![\mathtt{p}]\!]\,(0) = 0 \,\}$$
$$B = \{\, \mathtt{p} \;:\; [\![\mathtt{p}]\!]\,(0) \notin \{0, \perp\} \,\}$$

recursively inseparable

First Result

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Example

$A = \{ \mathtt{p} \ : \ [\![\mathtt{p}]\!](0) = 0 \}$
$B = \{ \mathtt{p} \ : \ [\![\mathtt{p}]\!](0) \notin \{0, \perp\} \}$

recursively inseparable

Proof.

$\mathcal{P}$ decidable, partially extensional for $[\![\mathtt{p}]\!]$, $\mathcal{P}$ contains no program for $[\![\mathtt{q}]\!]$.

```
r'(x) = if r(0)=0 then p(x) else q(x)
```

□

First Result

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Example

$A = \{ \texttt{p} \ : \ [\![\texttt{p}]\!](0) = 0 \}$
$B = \{ \texttt{p} \ : \ [\![\texttt{p}]\!](0) \notin \{0, \bot\} \}$

recursively inseparable

Proof.

$\mathcal{P}$ decidable, partially extensional for $[\![\texttt{p}]\!]$, $\mathcal{P}$ contains no program for $[\![\texttt{q}]\!]$.

```
r'(x) = if r(0)=0 then p(x) else q(x)
```

$[\![\texttt{r}]\!](0) = 0 \Rightarrow \ \texttt{r'} \in \mathcal{P}$
$[\![\texttt{r}]\!](0) \notin \{0, \bot\} \Rightarrow \ \texttt{r'} \notin \mathcal{P}$ □

First Result

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Example

$A = \{\, \mathtt{p} \;:\; [\![\mathtt{p}]\!]\,(0) = 0 \,\}$
$B = \{\, \mathtt{p} \;:\; [\![\mathtt{p}]\!]\,(0) \notin \{0, \bot\} \,\}$

recursively inseparable

Proof.

$\mathcal{P}$ decidable, partially extensional for $[\![\mathtt{p}]\!]$, $\mathcal{P}$ contains no program for $[\![\mathtt{q}]\!]$.

```
r'(x) = if r(0)=0 then p(x) else q(x)
```

$[\![\mathtt{r}]\!]\,(0) = 0 \Rightarrow \; \mathtt{r'} \in \mathcal{P}$
$[\![\mathtt{r}]\!]\,(0) \notin \{0, \bot\} \Rightarrow \; \mathtt{r'} \notin \mathcal{P}$

recusively separated by $\mathcal{P}$. □

Examples

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Examples

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Example

A decidable set containing all programs for the identity also contains programs for constant functions, the infinite loop, sorting, SAT, deciding correctness of MELL proof nets, ...

Examples

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Example

A decidable set containing all programs for the identity also contains programs for constant functions, the infinite loop, sorting, SAT, deciding correctness of MELL proof nets, ...

Example (Rice)

Any non-empty extensional set is partially extensional. Hence, if decidable, must be extensionally universal, and thus trivial.

Examples

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Examples

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Example

Any computable function is computed by infinitely many programs: a finite set is decidable, hence if partially extensional would be extensionally universal.

Examples

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Example

Any computable function is computed by infinitely many programs: a finite set is decidable, hence if partially extensional would be extensionally universal.

Example

Any computable function is computed by programs of arbitrarily large size.

Example

Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

Example

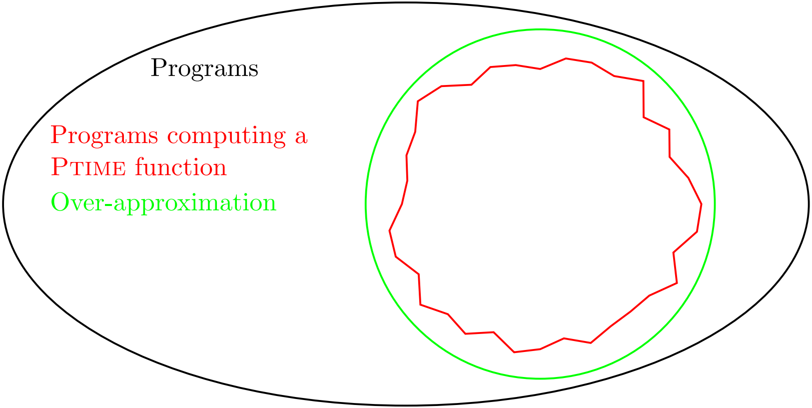
Theorem

Any non-empty, partially extensional and decidable set is extensionally universal.

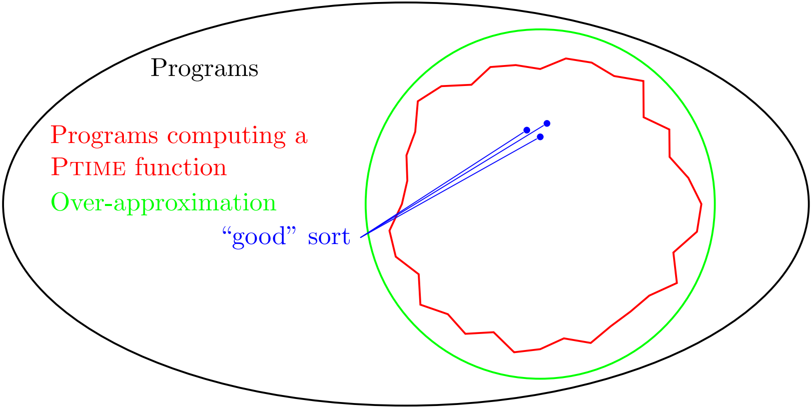
Example

Any decidable set containing all programs for PTIME functions contains programs for any computable function.

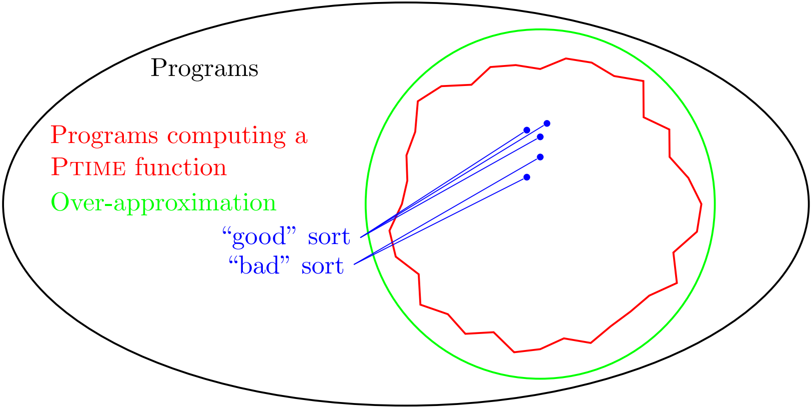
Example


Programs
Programs computing a PTIME function
Over-approximation

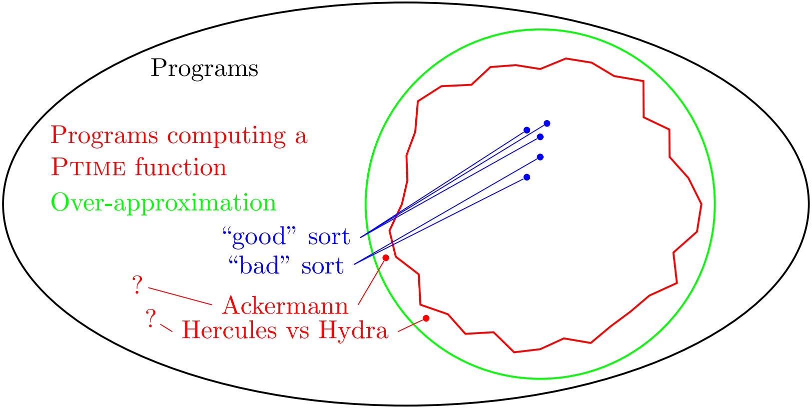
Example


Programs
Programs computing a PTIME function
Over-approximation
"good" sort

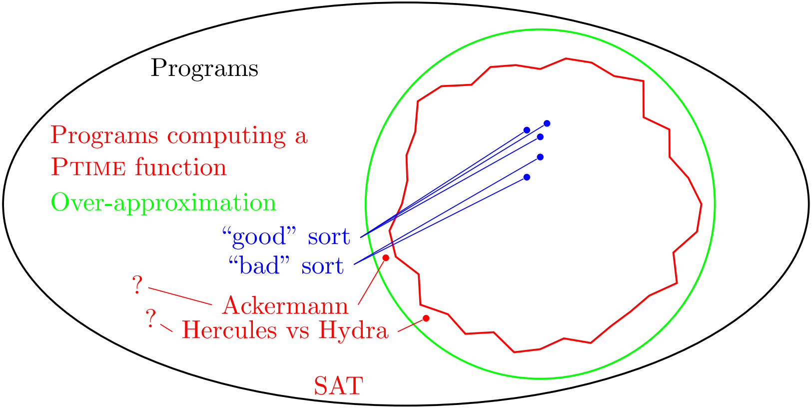
Example


Programs
Programs computing a
PTIME function
Over-approximation
"good" sort
"bad" sort

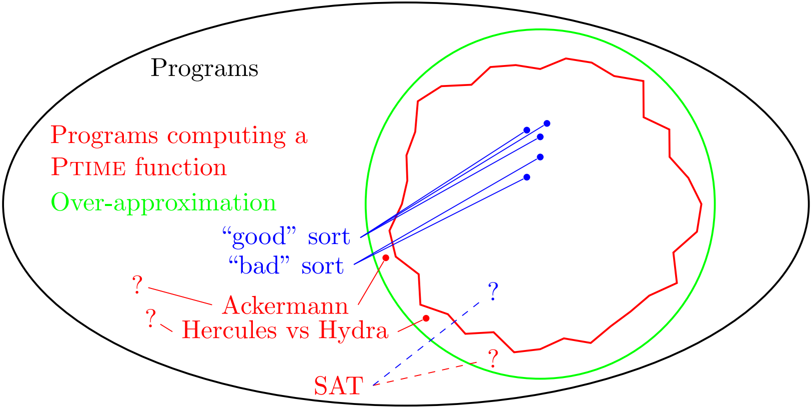
Example


Programs
Programs computing a
PTIME function
Over-approximation
"good" sort
"bad" sort
?
Ackermann
?
Hercules vs Hydra

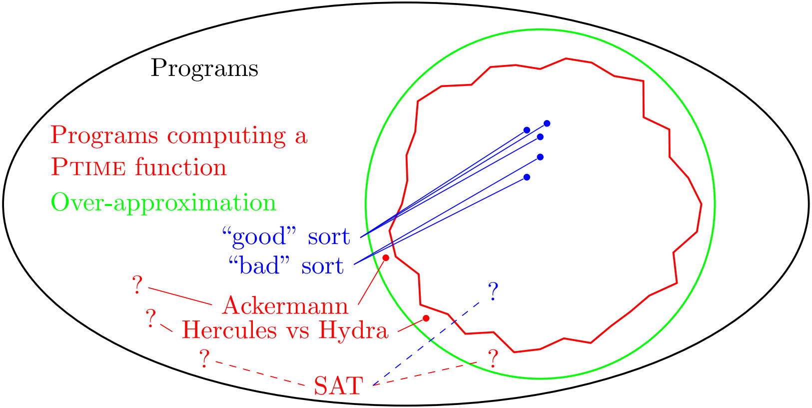
Example


Programs
Programs computing a
PTIME function
Over-approximation
"good" sort
"bad" sort
?
Ackermann
?
Hercules vs Hydra
SAT

Example


Programs
Programs computing a
PTIME function
Over-approximation
"good" sort
"bad" sort
?
Ackermann
?
Hercules vs Hydra
?
SAT
?

Example


Programs
Programs computing a PTIME function
Over-approximation
"good" sort
"bad" sort
?
Ackermann
?
Hercules vs Hydra
?
SAT
?
?

Second generalisation

Switching families

Definition

$(S, \approx)$: a set and an equivalence.
switching family compatible with $\approx$: a family $I = (\pi_s)_{s \in S}$ of computable total functions $\pi_s : S \times S \to S$

Switching families

Definition

$(S, \approx)$: a set and an equivalence.
switching family compatible with $\approx$: a family $I = (\pi_s)_{s \in S}$ of computable total functions $\pi_s : S \times S \to S$

$$\pi_s(x, y) \approx \begin{cases} x \\ y \\ ??? \end{cases}$$

Switching families

Definition

$(S, \approx)$: a set and an equivalence.
switching family compatible with $\approx$: a family $I = (\pi_s)_{s \in S}$ of computable total functions $\pi_s : S \times S \to S$

$$\pi_s(x, y) \approx \left\{ \begin{array}{l} x \\ y \\ ??? \end{array} \right\} \text{ for all or some } x, y.$$

Switching families

Definition

$(S, \approx)$: a set and an equivalence.
switching family compatible with $\approx$: a family $I = (\pi_s)_{s \in S}$ of computable total functions $\pi_s : S \times S \to S$

$$A_I = \{\, s \in S \;:\; \forall x, y. \pi_s(x,y) \approx x \,\}$$
$$B_I = \{\, s \in S \;:\; \forall x, y. \pi_s(x,y) \approx y \,\}$$

$$\pi_s(x,y) \approx \left\{ \begin{array}{l} x \\ y \\ ??? \end{array} \right) \text{ for all or some } x, y.$$

Switching families

Definition

$(S, \approx)$: a set and an equivalence.

switching family compatible with $\approx$: a family $I = (\pi_s)_{s \in S}$ of computable total functions $\pi_s : S \times S \to S$

$$\left.\begin{array}{l} A_I = \{\, s \in S \;:\; \forall x, y. \pi_s(x, y) \approx x \,\} \\ B_I = \{\, s \in S \;:\; \forall x, y. \pi_s(x, y) \approx y \,\} \end{array}\right. \quad \text{recursively inseparable.}$$

$$\pi_s(x, y) \approx \left\{ \begin{array}{l} x \\ y \\ ??? \end{array} \right\} \text{ for all or some } x, y.$$

Switching families

Definition

$(S, \approx)$: a set and an equivalence.

switching family compatible with $\approx$: a family $I = (\pi_s)_{s \in S}$ of computable total functions $\pi_s : S \times S \to S$

$A_I = \{ s \in S \ : \ \forall x, y. \pi_s(x, y) \approx x \}$
$B_I = \{ s \in S \ : \ \forall x, y. \pi_s(x, y) \approx y \}$
recursively inseparable.

Example

Projections can form a switching family.

Switching families

Definition

$(S, \approx)$: a set and an equivalence.
switching family compatible with $\approx$: a family $I = (\pi_s)_{s \in S}$ of computable total functions $\pi_s : S \times S \to S$

$$\left.\begin{array}{l} A_I = \{\, s \in S \;:\; \forall x, y. \pi_s(x, y) \approx x \,\} \\ B_I = \{\, s \in S \;:\; \forall x, y. \pi_s(x, y) \approx y \,\} \end{array}\right. \quad \text{recursively inseparable.}$$

Example

Projections can form a switching family.

Example (Standard switching family)

$\mathtt{r}'(x) = \pi_{\mathtt{r}}(\mathtt{p}, \mathtt{q})(x) =$ `if r(0)=0 then p(x) else q(x)`.
Compatible with $\mathfrak{R}$ (and many others).

Vocabulary

$\mathfrak{P}$: equivalence on programs. A set of programs is:

- ~~extensional~~ *compatible* if it is the union of blocks of $\mathfrak{P}$;
- ~~partially extensional~~ *partially compatible* if it contains one block of $\mathfrak{P}$;
- ~~extensionally complete~~ *complete* (for a set of blocks) if it intersects each of these;
- ~~extensionally sound~~
- ~~an ICC characterisation~~
- ~~extensionally universal~~ *universal* if it interesects each single block of $\mathfrak{P}$.

Second Result

Theorem

Let $\mathfrak{P}$ be a partition of a set S and $I = (\pi_s)_{s \in S}$ be a switching family compatible with it.
Any non-empty decidable partially compatible subset of S is universal.

Second Result

Theorem

Let $\mathfrak{P}$ be a partition of a set S and $I = (\pi_s)_{s \in S}$ be a switching family compatible with it.
Any non-empty decidable partially compatible subset of S is universal.

Proof.

$[x] \subset S'$, $[y]$ $\quad S' = \emptyset \qquad s' = \pi_s(x, y)$

$\pi_s(x, y)\mathfrak{P}x \Rightarrow s' \in S'$
$\pi_s(x, y)\mathfrak{P}y \Rightarrow s' \notin S'$ recursively inseparable. □

Example (1)

Theorem

Any non-empty decidable partially compatible set of programs is universal.

Example (1)

Theorem

Any non-empty decidable partially compatible set of programs is universal.

Example (Complexity)

Φ: complexity measure (Blum). $\mathtt{p} \equiv_\Phi \mathtt{q}$ iff $\Phi_\mathtt{p} \in \Theta(\Phi_\mathtt{q})$.

The standard switching family is compatible with $\equiv_\Phi$.
$\mathtt{r}'(x) = \pi_\mathtt{r}(\mathtt{p}, \mathtt{q})(x) =$ `if r(0)=0 then p(x) else q(x)`.
when `r(0)` terminates it does so with a constant complexity.

Any non-empty decidable set of programs partially compatible with $\equiv_\Phi$ is universal and must contain programs of arbitrarily high complexity.

Example (2)

Theorem

Any non-empty decidable partially compatible set of programs is universal.

Example (2)

Theorem

Any non-empty decidable partially compatible set of programs is universal.

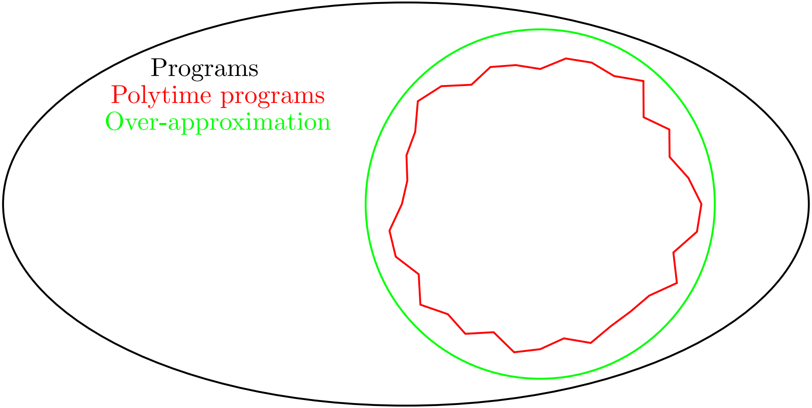
Example (Polynomial time)

Φ: time complexity. PPTIME: set of polytime *programs* (**not** all programs computing PTIME functions); it is undecidable and partially compatible with $\equiv_\Phi$.

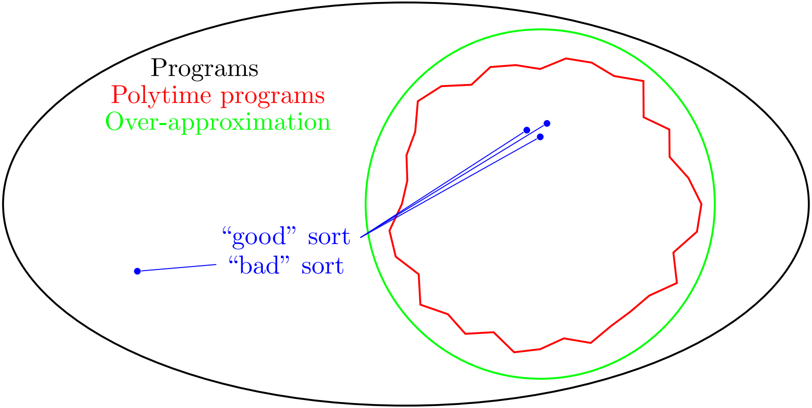
Any decidable set of programs including all polytime programs also includes programs of arbitrarily high time complexity.

Any attempt at finding a decidable over-approximation of PPTIME is doomed to also contain many extremely "bad" programs.

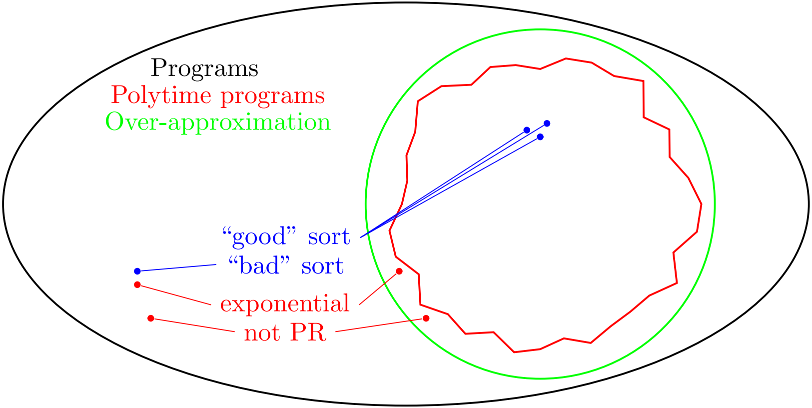
Example (2)



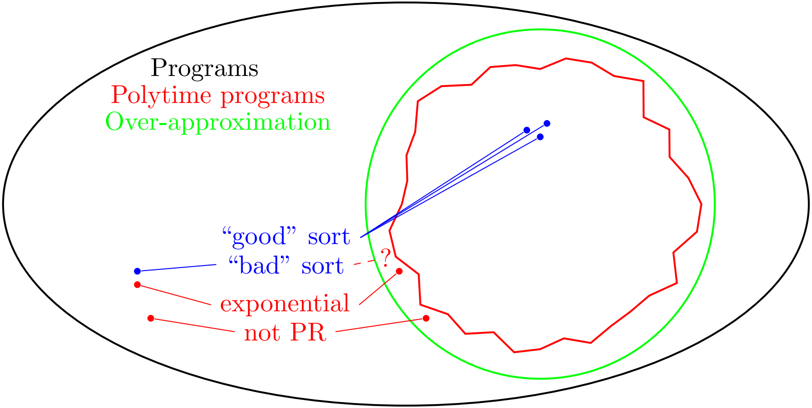
Example (2)


Programs
Polytime programs
Over-approximation
"good" sort
"bad" sort

Example (2)


Programs
Polytime programs
Over-approximation
"good" sort
"bad" sort
exponential
not PR

Example (2)


Programs
Polytime programs
Over-approximation
"good" sort
"bad" sort
?
exponential
not PR

Example (3)

Theorem

Any non-empty decidable partially compatible set of programs is universal.

Example (3)

Theorem

Any non-empty decidable partially compatible set of programs is universal.

Example (Linear space (not closed under composition))

Φ: space complexity. PLinSpace: set of *programs* computing in linear space; it is partially compatible with $\equiv_\Phi$.

Any decidable set of programs including all linear space programs also contains programs of arbitrarily high space complexity.

Example (Asperti-Rice)

Theorem

Any non-empty decidable partially compatible set of programs is universal.

Example (Asperti-Rice)

Theorem

Any non-empty decidable partially compatible set of programs is universal.

Example (Asperti-Rice)

The standard switching family is compatible with $\mathfrak{A} = \mathfrak{R}^{\top} \equiv_{\Phi}$.

Any decidable non-empty set partially compatible with $\mathfrak{A}$ is universal.

Especially, the only decidable unions of blocks of $\mathfrak{A}$ are the trivial ones.

Going further

Example (Spambot)

$\texttt{p} \equiv \texttt{q}$ if they send the same number of mails (**not** a Blum complexity measure). The standard switching family is compatible with it.

Any decidable set containing all the programs that never send mail also contains spambots.

Going further

Example (Spambot)

$\texttt{p} \equiv \texttt{q}$ if they send the same number of mails (**not** a Blum complexity measure). The standard switching family is compatible with it.

Any decidable set containing all the programs that never send mail also contains spambots.

Other equivalences?

Going further

Example (Spambot)

$\mathtt{p} \equiv \mathtt{q}$ if they send the same number of mails (**not** a Blum complexity measure). The standard switching family is compatible with it.

Any decidable set containing all the programs that never send mail also contains spambots.

Other equivalences?

Other switching families?