Ramsey vs. lexicographic termination proving

Byron Cook
Abigail See
Florian Zuleger

Terminator

Terminator proves termination using:

- Iterative algorithm
- Ramsey-based termination arguments

Terminator

Terminator proves termination using:

- Iterative algorithm
- Ramsey-based termination arguments

Question: Can we use the iterative algorithm *without* using Ramsey-based termination arguments?

Answer: Yes, and it's much faster

Terminator

Terminator proves termination using:

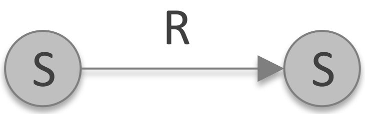
- Iterative algorithm
- ~~Ramsey-based termination arguments~~
  Lexicographic termination arguments

Question: Can we use the iterative algorithm *without* using Ramsey-based termination arguments?

Answer: Yes, and it's much faster

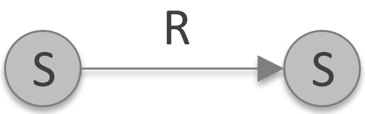
Proving termination

- A *program* $P = (S, R)$
  - Set of states S
  - Transition relation $R \subseteq S \times S$



Proving termination

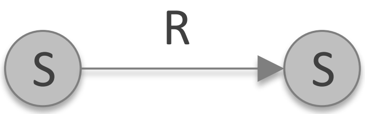
- A *program* $P = (S, R)$
  - Set of states S
  - Transition relation $R \subseteq S \times S$



- We want to prove that *R* is *well-founded*, i.e. doesn't contain infinite sequences

Proving termination

- A *program* $P = (S, R)$
  - Set of states S
  - Transition relation $R \subseteq S \times S$



- We want to prove that *R* is *well-founded*, i.e. doesn't contain infinite sequences

- *R* is well-founded $\Longleftrightarrow$ *P* terminates

Proving termination

- A *program* $P = (S, R)$
  - Set of states S
  - Transition relation $R \subseteq S \times S$

- We want to prove that R is *well-founded*, i.e. doesn't contain infinite sequences

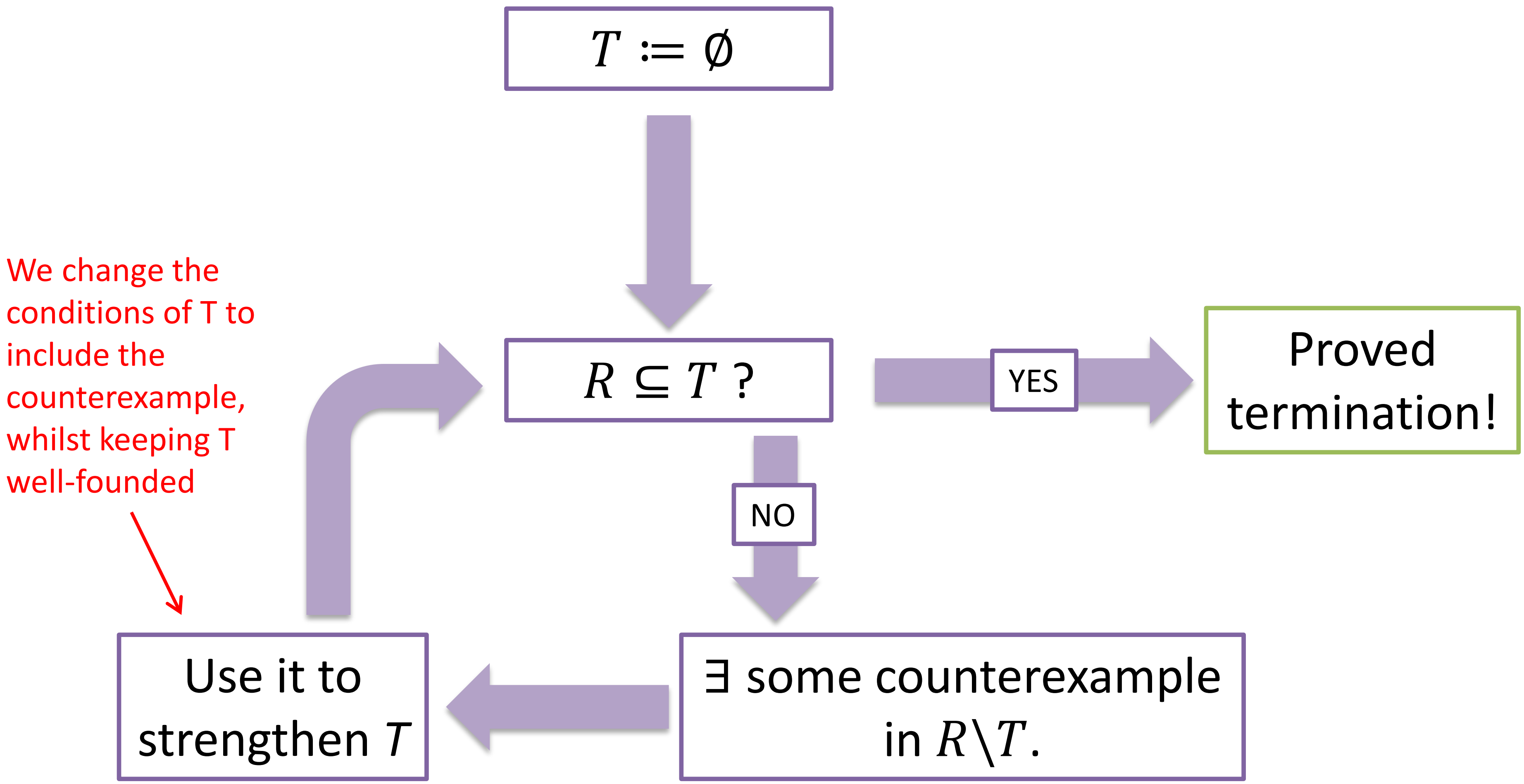
- R is well-founded $\Leftrightarrow$ P terminates

Usually a *condition* that must be met by all transitions in R

- **Aim**: find a well-founded relation T (the *termination argument*) such that $R \subseteq T$

Iteratively constructing T

Aim: find well-founded T such that $R \subseteq T$.



Ranking functions

- A **ranking function** is a function $f: S \longmapsto \mathbb{N}$ (or any well-ordered set)

Ranking functions

- A **ranking function** is a function $f: S \longmapsto \mathbb{N}$ (or any well-ordered set)
- We use them to construct termination arguments

Ranking functions

- A **ranking function** is a function $f: S \longmapsto \mathbb{N}$ (or any well-ordered set)

- We use them to construct termination arguments

- e.g. $T_f = \{(s,t) | \; f(s) > f(t) \;\; \wedge \;\; f(s) > 0\}$

"f decreases and is bounded below by 0"

Ranking functions

- A **ranking function** is a function $f: S \longmapsto \mathbb{N}$ (or any well-ordered set)

- We use them to construct termination arguments

- e.g. $T_f = \{(s,t) | \ f(s) > f(t) \ \wedge \ f(s) > 0\}$

"f decreases and is bounded below by 0"

- This is well-founded, so if $R \subseteq T_f$ then we have proved termination.

Ranking functions

- A **ranking function** is a function $f: S \longmapsto \mathbb{N}$ (or any well-ordered set)
- We use them to construct termination arguments
- e.g. $T_f = \{(s,t) | \, f(s) > f(t) \;\; \wedge \;\; f(s) > 0\}$

  "f decreases and is bounded below by 0"
- This is well-founded, so if $R \subseteq T_f$ then we have proved termination.
- However it is often difficult or impossible to find such a ranking function.

Ramsey-based termination arguments

- We use *several* ranking functions $\{f_1, f_2, \dots, f_n\}$ to construct *T:*

$$T = T_{f_1} \cup T_{f_2} \cup \cdots \cup T_{f_n}$$

Ramsey-based termination arguments

- We use *several* ranking functions $\{f_1, f_2, \ldots, f_n\}$ to construct *T:*

$$T = T_{f_1} \cup T_{f_2} \cup \cdots \cup T_{f_n}$$

- This condition says "at least one of $\{f_1, f_2, \ldots, f_n\}$ decreases towards 0"

Ramsey-based termination arguments

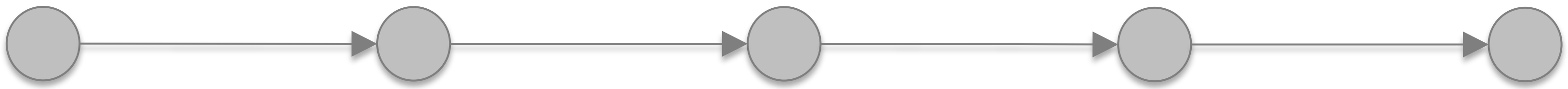
- We use *several* ranking functions $\{f_1, f_2, \ldots, f_n\}$ to construct *T:*

$$T = T_{f_1} \cup T_{f_2} \cup \cdots \cup T_{f_n}$$

- This condition says "at least one of $\{f_1, f_2, \ldots, f_n\}$ decreases towards 0"

The *transitive closure* of *R*

- Unfortunately we must prove $R^+ \subseteq T$ to prove *P* terminates.



Ramsey-based termination arguments

- We use *several* ranking functions $\{f_1, f_2, \ldots, f_n\}$ to construct *T:*

$$T = T_{f_1} \cup T_{f_2} \cup \cdots \cup T_{f_n}$$

- This condition says "at least one of $\{f_1, f_2, \ldots, f_n\}$ decreases towards 0"

The *transitive closure* of *R*

- Unfortunately we must prove $R^+ \subseteq T$ to prove *P* terminates.

- The proof that this is a sufficient condition uses *Ramsey's Theorem*

Ramsey-based termination arguments

- We use *several* ranking functions $\{f_1, f_2, \dots, f_n\}$ to construct *T:*

$$T = T_{f_1} \cup T_{f_2} \cup \cdots \cup T_{f_n}$$

- This condition says "at least one of $\{f_1, f_2, \dots, f_n\}$ decreases towards 0"

The *transitive closure* of *R*

- Unfortunately we must prove $R^+ \subseteq T$ to prove *P* terminates.

- The proof that this is a sufficient condition uses *Ramsey's Theorem*

- So *T* is a **Ramsey-based termination argument**.

Lexicographic termination arguments

- Put the ranking functions in some **order** $\langle\, f_1, f_2, \ldots, f_n \,\rangle$

Lexicographic termination arguments

- Put the ranking functions in some **order** $\langle f_1, f_2, \ldots, f_n \rangle$
- The condition of *T*: "at least one of $\langle f_1, f_2, \ldots, f_n \rangle$ decreases towards 0, *and the preceding ranking functions do not increase*"

Lexicographic termination arguments

- Put the ranking functions in some **order** $\langle f_1, f_2, \ldots, f_n \rangle$
- The condition of *T*: "at least one of $\langle f_1, f_2, \ldots, f_n \rangle$ decreases towards 0, *and the preceding ranking functions do not increase*"

e.g. $$\langle f_1, f_2, f_3, f_4, f_5 \rangle$$

Lexicographic termination arguments

- Put the ranking functions in some **order** $\langle f_1, f_2, \ldots, f_n \rangle$
- The condition of *T*: "at least one of $\langle f_1, f_2, \ldots, f_n \rangle$ decreases towards 0, *and the preceding ranking functions do not increase*"

e.g.

$$\langle \underset{\substack{\downarrow \\ \mathbf{0}}}{f_1}, f_2, f_3, f_4, f_5 \rangle$$

Lexicographic termination arguments

- Put the ranking functions in some **order** $\langle\, f_1, f_2, \ldots, f_n \,\rangle$

- The condition of *T*: "at least one of $\langle\, f_1, f_2, \ldots, f_n \,\rangle$ decreases towards 0, *and the preceding ranking functions do not increase*"

e.g. $$\langle\, \overset{\not\uparrow}{f_1}, \underset{\substack{\downarrow \\ \mathbf{0}}}{f_2}, f_3, f_4, f_5 \,\rangle$$

Lexicographic termination arguments

- Put the ranking functions in some **order** $\langle\, f_1, f_2, \ldots, f_n \,\rangle$
- The condition of *T*: "at least one of $\langle\, f_1, f_2, \ldots, f_n \,\rangle$ decreases towards 0, *and the preceding ranking functions do not increase*"

e.g.

$$\langle\, \overset{\not\uparrow}{f_1},\ \overset{\not\uparrow}{f_2},\ \underset{\substack{\downarrow \\ \mathbf{0}}}{f_3},\ f_4,\ f_5 \,\rangle$$

Lexicographic termination arguments

- Put the ranking functions in some **order** $\langle\, f_1, f_2, \ldots, f_n \,\rangle$

- The condition of *T*: “at least one of $\langle\, f_1, f_2, \ldots, f_n \,\rangle$ decreases towards 0, *and the preceding ranking functions do not increase*”

e.g. $$\langle\, f_1, f_2, f_3, f_4, f_5 \,\rangle$$

0

Lexicographic termination arguments

- Put the ranking functions in some **order** $\langle f_1, f_2, \dots, f_n \rangle$
- The condition of *T*: "at least one of $\langle f_1, f_2, \dots, f_n \rangle$ decreases towards 0, *and the preceding ranking functions do not increase*"

e.g. $\langle f_1, f_2, f_3, f_4, f_5 \rangle$

0

Lexicographic termination arguments

- Put the ranking functions in some **order** $\langle f_1, f_2, \dots, f_n \rangle$
- The condition of *T*: "at least one of $\langle f_1, f_2, \dots, f_n \rangle$ decreases towards 0, *and the preceding ranking functions do not increase*"
- This is a **lexicographic termination argument**.

Lexicographic termination arguments

- Put the ranking functions in some **order** $\langle f_1, f_2, \ldots, f_n \rangle$

- The condition of *T*: "at least one of $\langle f_1, f_2, \ldots, f_n \rangle$ decreases towards 0, *and the preceding ranking functions do not increase*"

- This is a **lexicographic termination argument**.

- Suffices to prove $R \subseteq T$ to prove termination.
  (No need to consider R^+)

Ramsey vs. lexicographic termination arguments

Ramsey

$\{f_1, f_2, \dots, f_n\}$

$R^+ \subseteq T$

"at least one of the RFs decreases"

Lexicographic

$\langle f_1, f_2, \dots, f_n \rangle$

$R \subseteq T$

"at least one of the RFs decreases, *and* none of the preceding RFs increase"

Ramsey vs. lexicographic termination arguments

Ramsey

$\{f_1, f_2, \dots, f_n\}$

$R^+ \subseteq T$

"at least one of the RFs decreases"

Prove an easier condition for all **sequences** of transitions

Lexicographic

$\langle f_1, f_2, \dots, f_n \rangle$

$R \subseteq T$

"at least one of the RFs decreases, *and* none of the preceding RFs increase"

Prove a **stricter** condition for all **single** transitions

Overall faster to construct iteratively

Procedure to construct lexicographic termination arguments

- The counterexamples we find during the iterative algorithm are in the form of **cycles** (paths returning to the same program location).

Procedure to construct lexicographic termination arguments

- The counterexamples we find during the iterative algorithm are in the form of **cycles** (paths returning to the same program location).
- We represent them as **relations** on S, e.g.
  - cycle $\pi = "x := x - 1"$
  - relation $[\![\pi]\!] = \{(s, t) | t(x) = s(x) - 1\}$

Procedure to construct lexicographic termination arguments

- The counterexamples we find during the iterative algorithm are in the form of **cycles** (paths returning to the same program location).

- We represent them as **relations** on S, e.g.
  - cycle $\pi = "x := x - 1"$
  - relation $[\![\pi]\!] = \{(s,t) | t(x) = s(x) - 1\}$

- During each step of our iterative algorithm, we have the relations we've found so far, put in some order $\langle \rho_1, \ldots, \rho_n \rangle$.

Procedure to construct lexicographic termination arguments

- The counterexamples we find during the iterative algorithm are in the form of **cycles** (paths returning to the same program location).

- We represent them as **relations** on S, e.g.
  - cycle $\pi = "x \coloneqq x - 1"$
  - relation $[\![\pi]\!] = \{(s,t) | t(x) = s(x) - 1\}$

- During each step of our iterative algorithm, we have the relations we've found so far, put in some order $\langle \rho_1, \dots, \rho_n \rangle$.

- We attempt to find a lexicographic ranking function $\langle f_1, \dots, f_n \rangle$ such that $\forall i, \rho_i$ decreases f_i towards 0 and does not increase any of $f_1, \dots, f_{i-1}$.

Procedure to construct lexicographic termination arguments

- The counterexamples we find during the iterative algorithm are in the form of **cycles** (paths returning to the same program location).
- We represent them as **relations** on S, e.g.
  - cycle $\pi = "x \coloneqq x - 1"$
  - relation $[\![\pi]\!] = \{(s,t) | t(x) = s(x) - 1\}$
- During each step of our iterative algorithm, we have the relations we've found so far, put in some order $\langle \rho_1, \dots, \rho_n \rangle$.
- We attempt to find a lexicographic ranking function $\langle f_1, \dots, f_n \rangle$ such that $\forall i, \rho_i$ decreases f_i towards 0 and does not increase any of $f_1, \dots, f_{i-1}$.
- Then $\rho_1 \cup \cdots \cup \rho_n \subseteq T$.

Procedure to construct lexicographic termination arguments

- The counterexamples we find during the iterative algorithm are in the form of **cycles** (paths returning to the same program location).

- We represent them as **relations** on S, e.g.
  - cycle $\pi = "x \coloneqq x - 1"$
  - relation $[\![\pi]\!] = \{(s,t) | t(x) = s(x) - 1\}$

- During each step of our iterative algorithm, we have the relations we've found so far, put in some order $\langle \rho_1, \ldots, \rho_n \rangle$.

- We attempt to find a lexicographic ranking function $\langle f_1, \ldots, f_n \rangle$ such that $\forall i, \rho_i$ decreases f_i towards 0 and does not increase any of $f_1, \ldots, f_{i-1}$.

- Then $\rho_1 \cup \cdots \cup \rho_n \subseteq T$.

- We keep adding relations ρ and functions f until (hopefully) $R \subseteq T$.

Procedure to construct lexicographic termination arguments

input: program P

T := $\emptyset$, empty termination argument
Π := $\langle\rangle$, empty sequence of relations

Stores all the cycles we've found so far

repeat

  if $\exists$ cycle π in P s.t. $[\![\pi]\!] \not\subseteq T$ **then**

Find a cycle that doesn't obey the termination arg.

    let $n = \text{length}(\Pi) = \text{length}\langle\rho_1, \rho_2, \ldots, \rho_n\rangle$
    for $i = 1$ to $n + 1$ do
      let $\Pi_i = \langle\rho_1, \rho_2, \ldots, \rho_{i-1}, [\![\pi]\!], \rho_i, \ldots, \rho_n\rangle$

Try inserting the new relation in all possible places

    if $\exists$ lex. ranking function $\langle f_1, f_2, \ldots, f_{n+1}\rangle$ for some Π_i **then**
      Π := Π_i
      T := lex. termination argument given by $\langle f_1, f_2, \ldots, f_{n+1}\rangle$

If we can find a lex. termination arg. for one of the orderings, let that be the new *T*

    else
        report "Unknown"

  else
    report "Success"

end.

Example

```
while x>0 && y>0 do
  if * then
    x := x - 1;
  else
    x := *
    y := y - 1;
  fi
done
```

Example

```
while x>0 && y>0 do
  if * then
    x := x - 1;
  else
    x := *
    y := y - 1;
  fi
done
```

$\rho_1 = x > 0 \land y > 0 \land x' = x - 1 \land y' = y$

$f_1 = x$

Example

```
while x>0 && y>0 do
  if * then
    x := x - 1;
  else
    x := *
    y := y - 1;
  fi
done
```

$\rho_1 = x > 0 \land y > 0 \land x' = x - 1 \land y' = y$

$f_1 = x$

$\Rightarrow T = T_{f_1}.\ R \subseteq T\ ?$

Example

```
while x>0 && y>0 do
  if * then
    x := x - 1;
  else
    x := *
    y := y - 1;
  fi
done
```

$\rho_1 = x > 0 \land y > 0 \land x' = x - 1 \land y' = y$

$f_1 = x$

$\Rightarrow T = T_{f_1}.\ R \subseteq T$?

No:

$\rho_2 = x > 0 \land y > 0 \land x' = * \land y' = y - 1$

$f_2 = y$

Example

```
while x>0 && y>0 do
  if * then
    x := x - 1;
  else
    x := *
    y := y - 1;
  fi
done
```

$\rho_1 = x > 0 \wedge y > 0 \wedge x' = x - 1 \wedge y' = y$

$f_1 = x$

$\Rightarrow T = T_{f_1}.\ R \subseteq T$?

No:

$\rho_2 = x > 0 \wedge y > 0 \wedge x' = * \wedge y' = y - 1$

$f_2 = y$

Valid if ρ_2 does not increase f_1

Valid if ρ_1 does not increase f_2

$\cancel{\langle f_1, f_2 \rangle}$ or $\langle f_2, f_1 \rangle$?

Example

```
while x>0 && y>0 do
  if * then
    x := x - 1;
  else
    x := *
    y := y - 1;
  fi
done
```

$\rho_1 = x > 0 \wedge y > 0 \wedge x' = x - 1 \wedge y' = y$

$f_1 = x$

$\Rightarrow T = T_{f_1}.\ R \subseteq T\,?$

No:

$\rho_2 = x > 0 \wedge y > 0 \wedge x' = * \wedge y' = y - 1$

$f_2 = y$

Valid if ρ_2 does not increase f_1

Valid if ρ_1 does not increase f_2

$\cancel{\langle f_1, f_2 \rangle}$ or $\langle f_2, f_1 \rangle$?

$R \subseteq T$?

"f_2 decreases towards 0, or f_1 decreases towards 0 and f_2 does not increase"

Example

```
while x>0 && y>0 do
  if * then
    x := x - 1;
  else
    x := *
    y := y - 1;
  fi
done
```

$\rho_1 = x > 0 \wedge y > 0 \wedge x' = x - 1 \wedge y' = y$

$f_1 = x$

$\Rightarrow T = T_{f_1}.\ R \subseteq T$?

No:

$\rho_2 = x > 0 \wedge y > 0 \wedge x' = * \wedge y' = y - 1$

$f_2 = y$

Valid if ρ_2 does not increase f_1

Valid if ρ_1 does not increase f_2

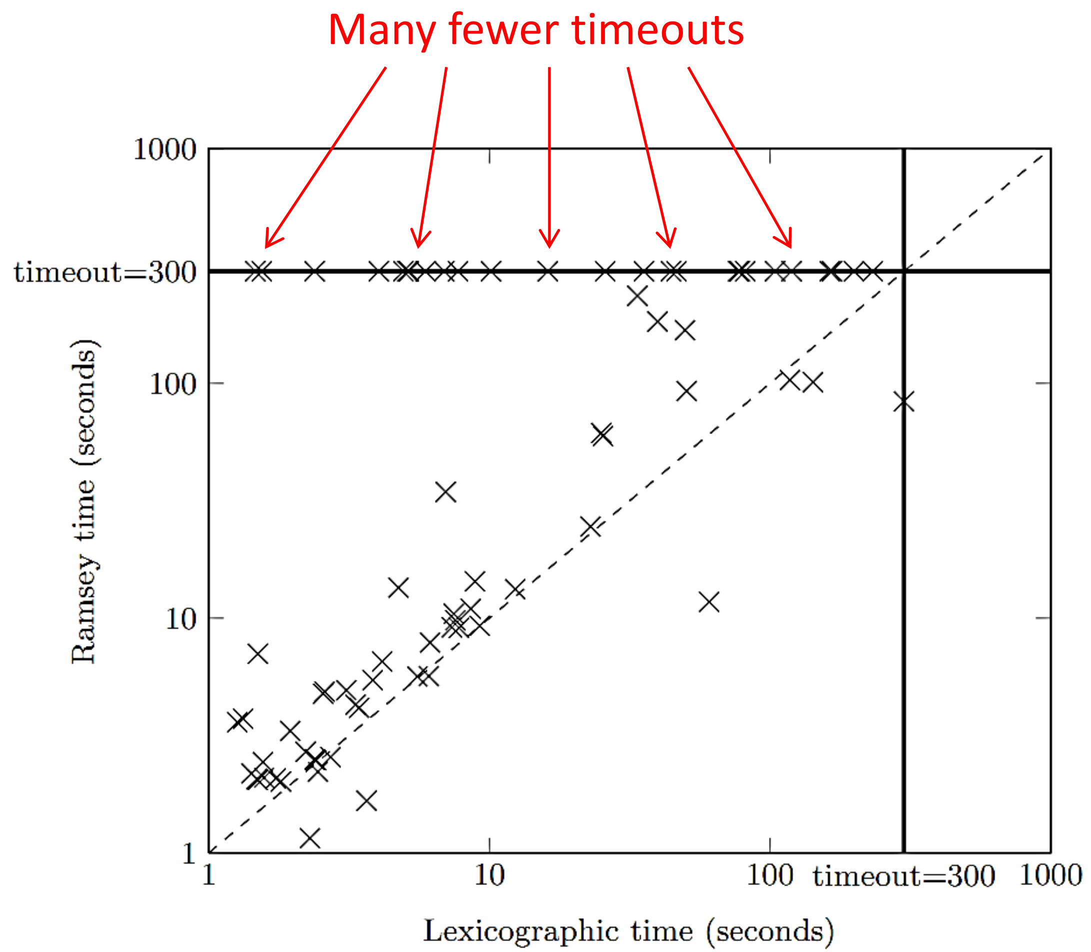
$\cancel{\langle f_1, f_2 \rangle}$ or $\langle f_2, f_1 \rangle$?

$R \subseteq T$?

"f_2 decreases towards 0, or f_1 decreases towards 0 and f_2 does not increase"

Yes: we have proved termination

Results


Many fewer timeouts
1000
timeout=300
100
10
1
Ramsey time (seconds)
1
10
100
timeout=300
1000
Lexicographic time (seconds)

A disadvantage of lexicographic termination arguments

- Existence of a Ramsey-based termination argument **does not imply** existence of a lexicographic termination argument.

A disadvantage of lexicographic termination arguments

- Existence of a Ramsey-based termination argument **does not imply** existence of a lexicographic termination argument.

- So occasionally we cannot find a lexicographic termination argument (when we can find a Ramsey one).

A disadvantage of lexicographic termination arguments

- Existence of a Ramsey-based termination argument **does not imply** existence of a lexicographic termination argument.
- So occasionally we cannot find a lexicographic termination argument (when we can find a Ramsey one).
- In our experience this is rare.

A tricky example

```
while x<>0 do
  if x>0 then
    x := x - 1;
  else
    x := x + 1;
  fi
done
```

$$f_1 = x$$

$$f_2 = -x$$

A tricky example

```
while x<>0 do
  if x>0 then
    x := x - 1;
  else
    x := x + 1;
  fi
done
```

$$f_1 = x$$

$$f_2 = -x$$

A tricky example

```
while x<>0 do
  if x>0 then
    x := x - 1;
  else
    x := x + 1;
  fi
done
```

$$f_1 = x$$

$$f_2 = -x$$

A tricky example

```
while x<>0 do
  if x>0 then
    x := x - 1;
  else
    x := x + 1;
  fi
done
```

$$f_1 = x$$

$$f_2 = -x$$

$T_{f_1} \cup T_{f_2}$ is a valid Ramsey-based termination argument.

A tricky example

```
while x<>0 do
  if x>0 then
    x := x - 1;
  else
    x := x + 1;
  fi
done
```

$f_1 = x$

$f_2 = -x$

if one decreases, the other must increase!

$T_{f_1} \cup T_{f_2}$ is a valid Ramsey-based termination argument.

$\langle f_1, f_2 \rangle$? ✗

$\langle f_2, f_1 \rangle$? ✗

A tricky example

```
while x<>0 do
  if x>0 then
    x := x - 1;
  else
    x := x + 1;
  fi
done
```

$f_1 = x$

$f_2 = -x$

if one decreases, the other must increase!

$T_{f_1} \cup T_{f_2}$ is a valid Ramsey-based termination argument.

$\langle f_1, f_2 \rangle$? ❌

$\langle f_2, f_1 \rangle$? ❌

No (linear) lexicographic termination argument.

Solution

```
c := 0
while x<>0
  if x>0 then
    if c=0 then
      c := 1
    x := x - 1;
  else
    if c=0 then
      c := 2
    x := x + 1;
```

Prove termination separately for c=1 and c=2, i.e. have different termination arguments for c=1 and c=2:

$\langle f_1 \rangle = \langle x \rangle$ for c=1
$\langle f_2 \rangle = \langle -x \rangle$ for c=2

Solution

```
c := 0
while x<>0
  if x>0 then
    if c=0 then
      c := 1
    x := x - 1;
  else
    if c=0 then
      c := 2
    x := x + 1;
```

Prove termination separately for c=1 and c=2, i.e. have different termination arguments for c=1 and c=2:

$\langle f_1 \rangle = \langle x \rangle$ for c=1
$\langle f_2 \rangle = \langle -x \rangle$ for c=2

This solution deals with cases where there is a split case into several disjoint programs.

Conclusion

- Using lexicographic instead of Ramsey-based termination arguments is much faster in an iterative termination-proving algorithm such as Terminator's.
- Occasionally we can't find lexicographic termination arguments, but there are some tricks to get around this.

Thank you for listening

Any questions?