Ramsey vs. lexicographic termination proving

Byron Cook
Abigail See
Florian Zuleger
Terminator proves termination using:

- Iterative algorithm
- Ramsey-based termination arguments
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 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
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 $\iff P$ 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 \( \iff \) \( P \) terminates

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

Usually a condition that must be met by all transitions in \( R \)
Aim: find well-founded $T$ such that $R \subseteq T$.

1. $T := \emptyset$

2. $R \subseteq T$ ?

   YES

   Proved termination!

   NO

   $\exists$ some counterexample in $R \setminus T$.

   Use it to strengthen $T$.

   We change the conditions of $T$ to include the counterexample, whilst keeping $T$ well-founded.
A ranking function is a function $f : S \mapsto \mathbb{N}$ (or any well-ordered set)
Ranking functions

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

• We use them to construct termination arguments
Ranking functions

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

• We use them to construct termination arguments

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

“$f$ decreases and is bounded below by 0”
**Ranking functions**

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

- We use them to construct termination arguments

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

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

“$f$ decreases and is bounded below by 0”
A ranking function is a function $f : S \mapsto \mathbb{N}$ (or any well-ordered set).

We use them to construct termination arguments.

For example, $T_f = \{(s, t) | f(s) > f(t) \land f(s) > 0\}$

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 \( \{f_1, f_2, \ldots, f_n\} \) 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”

- Unfortunately we must prove \( R^+ \subseteq T \) to prove \( P \) terminates.

The transitive closure of \( R \)
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”

• 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 \( \{f_1, f_2, \ldots, f_n\} \) 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”

• 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”

\[ \langle f_1, f_2, f_3, f_4, f_5 \rangle \]

\[ \downarrow \]

0
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”

\[
\langle f_1, f_2, f_3, f_4, f_5 \rangle
\]

e.g. 0
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”

\[
\langle f_1, f_2, f_3, f_4, f_5 \rangle
\]

e.g. 0
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”

\[ \langle f_1, f_2, f_3, f_4, f_5 \rangle \]
\[ 0 \]
e.g.
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.
Lexicographic termination arguments

• Put the ranking functions in some order \( f_1, f_2, \ldots, f_n \)

• The condition of \( T \): “at least one of \( f_1, f_2, \ldots, f_n \) 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

<table>
<thead>
<tr>
<th>Ramsey</th>
<th>Lexicographic</th>
</tr>
</thead>
<tbody>
<tr>
<td>({f_1, f_2, \ldots, f_n})</td>
<td>(\langle f_1, f_2, \ldots, f_n \rangle)</td>
</tr>
<tr>
<td>(R^+ \subseteq T)</td>
<td>(R \subseteq T)</td>
</tr>
<tr>
<td>“at least one of the RFs decreases”</td>
<td>“at least one of the RFs decreases, and none of the preceding RFs increase”</td>
</tr>
</tbody>
</table>

\(R^+ \subseteq T\) means that at least one of the Ramsey functions decreases, whereas \(R \subseteq T\) means that at least one of the lexicographic functions decreases, and none of the preceding functions increase.
Ramsey vs. lexicographic termination arguments

**Ramsey**

\[ \{f_1, f_2, \ldots, f_n\} \]

\[ R^+ \subseteq T \]

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

**Lexicographic**

\[ \langle f_1, f_2, \ldots, f_n \rangle \]

\[ R \subseteq T \]

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

“at least one of the RFs decreases, and none of the preceding RFs increase”

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 $\llbracket \pi \rrbracket = \{(s, t) | t(x) = s(x) - 1\}$

• During each step of our iterative algorithm, we have the relations we’ve found so far, put in some order $\langle \rho_1, ..., \rho_n \rangle$. 
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 := x - 1"
  - relation $\llbracket \pi \rrbracket = \{(s, t) | t(x) = s(x) - 1\}$

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

- We attempt to find a lexicographic ranking function $\langle f_1, ..., f_n \rangle$ such that $\forall i, \rho_i$ decreases $f_i$ towards 0 and does not increase any of $f_1, ..., f_{i-1}$. 
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 \( \pi = "x := x - 1" \)
  - Relation \( \llbracket \pi \rrbracket = \{(s, t) | t(x) = s(x) - 1\} \)

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

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

- 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 $\pi = "x := x - 1"$
  • 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 $\langle \rho_1, ..., \rho_n \rangle$.

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

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

• We keep adding relations $\rho$ and functions $f$ until (hopefully) $R \subseteq T$. 
Procedure to construct lexicographic termination arguments

input: program $P$

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

repeat

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

let $n = \text{length}(\Pi) = \text{length}(\langle \rho_1, \rho_2, \ldots, \rho_n \rangle)$

for $i = 1$ to $n + 1$ do

let $\Pi_i = \langle \rho_1, \rho_2, \ldots, \rho_i-1, [\pi], \rho_i, \ldots, \rho_n \rangle$

if $\exists$ lex. ranking function $\langle f_1, f_2, \ldots, f_{n+1} \rangle$ for some $\Pi_i$ then

$\Pi := \Pi_i$

$T := \text{lex. termination argument given by } \langle f_1, f_2, \ldots, f_{n+1} \rangle$

else

report “Unknown”

else

report “Success”

end.
Example

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

\[ \rho_1 = x > 0 \land y > 0 \land x' = x - 1 \land y' = y \]

\[ f_1 = x \]
\[ \rho_1 = x > 0 \land y > 0 \land x' = x - 1 \land y' = y \]
\[ f_1 = x \]
\[ \Rightarrow T = T_{f_1} \land R \subseteq T ? \]
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

\[ \rho_1 = x > 0 \land y > 0 \land x' = x - 1 \land y' = y \]

\[ f_1 = x \]

\[ \Rightarrow T = T_{f_1}. \quad R \subseteq T? \]

No:
\[ \rho_2 = x > 0 \land y > 0 \land x' = * \land y' = y - 1 \]

\[ f_2 = y \]

Valid if \( \rho_2 \) does not increase \( f_1 \)

Valid if \( \rho_1 \) does not increase \( f_2 \)

\[ \langle f_1, f_2 \rangle \text{ or } \langle f_2, f_1 \rangle? \]
Example

\[ \text{while } x > 0 \land y > 0 \text{ do} \]
\[ \quad \text{if } * \text{ then } \]
\[ \quad \quad x := x - 1; \]
\[ \quad \text{else } \]
\[ \quad \quad x := * \]
\[ \quad \quad y := y - 1; \]
\[ \text{fi} \]
\[ \text{done} \]

\[ \rho_1 = x > 0 \land y > 0 \land x' = x - 1 \land y' = y \]
\[ f_1 = x \]
\[ \Rightarrow T = T_{f_1}. \ R \subseteq T? \]

No:
\[ \rho_2 = x > 0 \land y > 0 \land x' = * \land y' = y - 1 \]
\[ f_2 = y \]

Valid if \( \rho_2 \) does not increase \( f_1 \)

Valid if \( \rho_1 \) does not increase \( f_2 \)

\langle f_1, f_2 \rangle \text{ 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

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

$f_1 = x$

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

Valid if $\rho_2$ does not increase $f_1$

No:

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

$f_2 = y$

Valid if $\rho_1$ does not increase $f_2$

$\langle f_1, f_2 \rangle$ or $\langle f_2, f_1 \rangle$?

$R \subseteq T$?

Yes: we have proved termination

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

Many fewer timeouts
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.
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

\[
f_1 = x \\
f_2 = -x
\]

\begin{verbatim}
while x<>0 do
    if x>0 then
        x := x - 1;
    else
        x := x + 1;
    fi
done
\end{verbatim}
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

$f_1 = x$
$f_2 = -x$

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

$\langle f_1, f_2 \rangle$? $\times$

$\langle f_2, f_1 \rangle$? $\times$
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
            end if
            x := x + 1;
    else
        x := x - 1;
    end if
end while
```

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

\[
\langle f_1 \rangle = \langle x \rangle \text{ for } c=1
\]

\[
\langle f_2 \rangle = \langle -x \rangle \text{ for } c=2
\]
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;
    else
        if c=0 then
            c := 1
            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?