Senescent Ground Tree Rewrite Systems

Matthew Hague

Royal Holloway University of London
Abstract

We study a restriction of ground tree rewrite systems with state.
  o Based on scope-bounded pushdown systems.
  o When the control state changes, nodes grow old.
  o When a subtree is rewritten is it young again.
  o After a fixed age, nodes become fixed (cannot be rewritten).

We show:
  o Control state reachability is Ackermann-complete.
  o Reachability of a regular set of trees is undecidable.

Builds on work by A. W. Lin (MFCS 2012).
A configuration contains a control state and a tree.
Ground Tree Rewrite Systems with State

A configuration contains a control state and a tree.

Control state from finite set \( \{1, 2, 3\} \)
A configuration contains a control state and a tree.

Control state from finite set \{1, 2, 3\}

Node labels from finite ranked alphabet \{a : 2, b : 1, c : 0, d : 0\}
Transitions of the system are via rules.
Ground Tree Rewrite Systems with State

Transitions of the system are via rules. E.g.
Ground Tree Rewrite Systems with State

Transitions of the system are via rules. E.g.
Transitions of the system are via rules. E.g.
Ground Tree Rewrite Systems with State

Transitions of the system are via rules. E.g.
Transitions of the system are via rules. E.g.
Transitions of the system are via rules. E.g.
Ground Tree Rewrite Systems with State

Transitions of the system are via rules. E.g.

1. \[ \begin{array}{c}
         c \\
        / \  \\
      a   b \\
        \  / \\
         d  c
\end{array} \]

2. \[ \begin{array}{c}
         c \\
        / \  \\
      b   d \\
        \  / \\
         c  b \\
\end{array} \]

1. \[ \begin{array}{c}
         c \\
        / \  \\
      a   b \\
        \  / \\
         d  a
\end{array} \]

2. \[ \begin{array}{c}
         c \\
        / \  \\
      b   b \\
        \  / \\
         d  b \\
\end{array} \]
And so, the system progresses...
And so, the system progresses...
And so, the system progresses...
And so, the system progresses...
And so, the system progresses... 

Regular reachability:
And so, the system progresses...

Regular reachability:

- From a given initial configuration...
And so, the system progresses...

Regular reachability:

- From a given initial configuration...
- ...can a given control state be reached.
- ...with a tree from a regular set.
And so, the system progresses...

Control state reachability:

- From a given initial configuration...
- ...can a given control state be reached.
- ...we don’t restrict the final tree.
And so, the system progresses... 

Control state reachability:
- From a given initial configuration...
- ...can a given control state be reached.
- ...we don’t restrict the final tree.

Undecidable!
Control state reachability for a two-stack pushdown system is well known to be undecidable.
Control state reachability for a two-stack pushdown system is well known to be undecidable.

Directly modelled by a ground tree rewrite system with state.
Control state reachability for a two-stack pushdown system is well known to be undecidable.

But we can spawn threads!

Directly modelled by a ground tree rewrite system with state.
Control state reachability for a two-stack pushdown system is well known to be undecidable.

But we can spawn threads!

Directly modelled by a ground tree rewrite system with state.
Not quite – many restrictions of multi-stack pushdown systems are decidable.

- Remove the control state [Hofman et al.].
- Bounded context switches [Qadeer&Rehof].
- Phase bounding [La Torre et al.].
- Ordered stacks [Breveglieri et al., Atig et al.].
- Bounded languages [Esparza, Ganty, Majumdar].
- Asynchronous method calls [Sen&Viswanathan, Heußner et al.].
- Nested locks [Kahlon].
- Tree-width [Madhusudan&Parlato].
- Split-width [Cyriac et al.].
Scope Bounded Pushdown Systems

Introduced by La Torre & Napoli.

Runs proceed in rounds, where each system runs in turn.
Scope Bounded Pushdown Systems

Introduced by La Torre & Napoli.

Runs proceed in rounds, where each system runs in turn.
Scope Bounded Pushdown Systems

Introduced by La Torre & Napoli.

Runs proceed in rounds, where each system runs in turn.
Scope Bounded Pushdown Systems

Introduced by La Torre & Napoli.

Runs proceed in **rounds**, where each system runs in turn.
Scope Bounded Pushdown Systems

Introduced by La Torre & Napoli.

Runs proceed in rounds, where each system runs in turn.
Scope Bounded Pushdown Systems

Introduced by La Torre & Napoli.

Runs proceed in rounds, where each system runs in turn.
Scope Bounded Pushdown Systems

Introduced by La Torre & Napoli.

Runs proceed in rounds, where each system runs in turn.
Scope Bounded Pushdown Systems

Introduced by La Torre & Napoli.

Runs proceed in rounds, where each system runs in turn.
A scope bounded system has a fixed bound $k$.

Pop can only remove characters from $\leq k$ rounds earlier.
A scope bounded system has a fixed bound $k$.

Pop can only remove characters from $\leq k$ rounds earlier.
A scope bounded system has a fixed bound $k$. Information that is too old can no longer be accessed.

Pop can only remove characters from $\leq k$ rounds earlier.
A scope bounded system has a fixed bound $k$.

**Senescent Ground Tree Rewrite Systems**
Nodes that are too old can no longer be rewritten.

Pop can only remove characters from $\leq k$ rounds earlier.
A senescent ground tree rewrite system…

- …is a ground tree rewrite system with state,
- …with a fixed maximum age $k$,
- …each node of the tree has an age,
- …changing the control state ages the nodes,
- …rewriting subtrees results in fresh nodes,
- …nodes over age $k$ can no longer be rewritten.
Senescent Ground Tree Rewrite Systems

A senescent ground tree rewrite system...

- ...is a ground tree rewrite system with state,
- ...with a fixed maximum age \( k \),
- ...each node of the tree has an age,
- ...changing the control state ages the nodes,
- ...rewriting subtrees results in fresh nodes,
- ...nodes over age \( k \) can no longer be rewritten.

Note:

- **Not** a restriction on the number of changes to a node.
  - (c.f. Abdulla et al. CAV 2002).
- A restriction on how long a node is **unchanged**.
Assign each node an **age**.
We will use suggestive colors:  young  adult  old .
Initially all nodes are young.

Assign each node an **age**.
We will use suggestive colors: young adult old.
When changing the control state the nodes age.
Rewritten nodes are young again.

Assign each node an age.
We will use suggestive colors: young adult old.
When changing the control state the nodes age. Rewritten nodes are young again.

Assign each node an age. We will use suggestive colors: young adult old.
If not changing control state, nodes do not age. Rewritten nodes are young again.

Assign each node an age.
We will use suggestive colors: young adult old.
If not changing control state, nodes do not age. Rewritten nodes are young again.

Assign each node an age. We will use suggestive colors: young adult old.
If we update the control state again...

Assign each node an age.
We will use suggestive colors: young adult old.
If we update the control state again…

Assign each node an age.
We will use suggestive colors: young adult old.
If we update the control state again...

Assign each node an **age**.
We will use suggestive colors: **young** adult old.
**Theorem**

Control state reachability of senescent ground tree rewrite systems is decidable and Ackermann-complete.

Subsumes: scope-bounded pushdowns; reset Petri net coverability.
Results

Theorem
Control state reachability of senescent ground tree rewrite systems is decidable and Ackermann-complete.

Subsumes: scope-bounded pushdowns; reset Petri net coverability.

Theorem
Reachability of a given regular set of trees is undecidable.