Algorithmic Aspects of WQO (Well-Quasi-Ordering) Theory

Part II: Algorithmic Applications of WQOs

Sylvain Schmitz & Philippe Schnoebelen

LSV, CNRS & ENS Cachan

ESSLLI 2016, Bozen/Bolzano, Aug 22-26, 2016

Lecture notes & exercises available at

http://www.lsv.fr/~schmitz/teach/2016_esslli

If you missed Part I

$(X,\leqslant)$ is a well-quasi-ordering (a wqo) if any infinite sequence $x_0,x_1,x_2\ldots$ over X contains an increasing pair $x_i \leqslant x_j$ (for some $i<j$)

Examples.

1. $(\mathbb{N}^k,\leqslant_\times)$ is a wqo (Dickson's Lemma)
   where, e.g., $(3,2,1) \leqslant_\times (5,2,2)$ but $(1,2,3) \not\leqslant_\times (5,2,2)$
2. $(\Sigma^*,\leqslant_*)$ is a wqo (Higman's Lemma)
   where, e.g., $abc \leqslant_* bacbc$ but $cba \not\leqslant_* bacbc$

Intuition motivating this course:

Analyzing the complexity of algorithms based on WQO-theory

$\simeq$

Bounding the index j (in the increasing pair above)
as a function of some relevant parameters

IF YOU MISSED PART I

$(X, \leqslant)$ is a well-quasi-ordering (a wqo) if any infinite sequence $x_0, x_1, x_2 \ldots$ over X contains an increasing pair $x_i \leqslant x_j$ (for some $i < j$)

Examples.

1. $(\mathbb{N}^k, \leqslant_\times)$ is a wqo (Dickson's Lemma)
   where, e.g., $(3,2,1) \leqslant_\times (5,2,2)$ but $(1,2,3) \not\leqslant_\times (5,2,2)$

2. $(\Sigma^*, \leqslant_*)$ is a wqo (Higman's Lemma)
   where, e.g., $abc \leqslant_* bacbc$ but $cba \not\leqslant_* bacbc$

Intuition motivating this course:

Analyzing the complexity of algorithms based on WQO-theory

$\simeq$

Bounding the index j (in the increasing pair above)
as a function of some relevant parameters

Outline for Part II

- Well-structured transition systems (WSTS's) and their decision algorithms
- Termination proofs for programs
- Relevance logics and their decidability
- … Karp-Miller trees ?

All of these are actual examples of algorithms that terminate thanks to wqo-theoretical arguments

Question for Part III. terminate in how many steps exactly?

Outline for Part II

- Well-structured transition systems (WSTS's) and their decision algorithms
- Termination proofs for programs
- Relevance logics and their decidability
- … Karp-Miller trees ?

All of these are actual examples of algorithms that terminate thanks to wqo-theoretical arguments

Question for Part III. terminate in how many steps exactly?

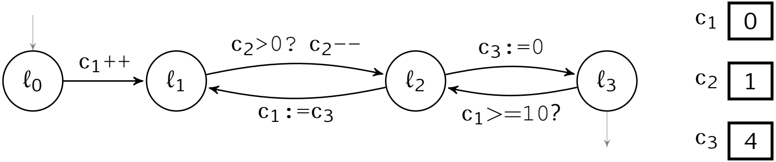
WSTS: Well-structured transition systems

In program verification, wqo's appear prominently under the guise of WSTS.

Def. A WSTS is a system $(S, \rightarrow, \leqslant)$ where

1. $(S, \rightarrow)$ with $\rightarrow \subseteq S \times S$ is a transition system
2. the set of states $(S, \leqslant)$ is wqo, and
3. the transition relation is compatible with the ordering (also called "monotonic"): $s \rightarrow t$ and $s \leqslant s'$ imply $s' \rightarrow t'$ for some $t' \geqslant t$

Some WSTS's: Monotonic Counter Machines



A run of M: $(\ell_0,0,1,4) \to (\ell_1,1,1,4) \to (\ell_2,1,0,4) \to (\ell_3,1,0,0)$

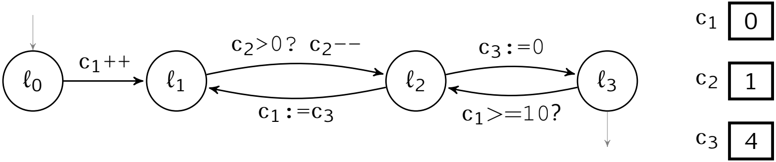
Ordering states: $(\ell_1,0,0,0) \leqslant (\ell_1,0,1,2)$ but $(\ell_1,0,0,0) \not\leqslant (\ell_2,0,1,2)$.
This is wqo as a product of wqo's: $(Loc,=) \times (\mathbb{N}^3,\leqslant_\times)$

Compatibility: easily checked when guards are upward-closed and assignments are monotonic functions of the variables.

NB. Other updates can be considered as long as they are monotonic. Extending guards require using a finer ordering.

Question. How does this compare to Minsky (counter) machines?

Some WSTS's: Monotonic Counter Machines



A run of M: $(\ell_0,0,1,4) \to (\ell_1,1,1,4) \to (\ell_2,1,0,4) \to (\ell_3,1,0,0)$

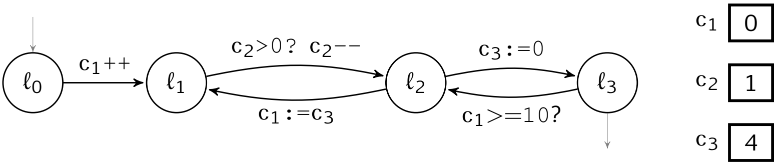
Ordering states: $(\ell_1,0,0,0) \leqslant (\ell_1,0,1,2)$ but $(\ell_1,0,0,0) \not\leqslant (\ell_2,0,1,2)$.
This is wqo as a product of wqo's: $(Loc,=) \times (\mathbb{N}^3,\leqslant_\times)$

Compatibility: easily checked when guards are upward-closed and assignments are monotonic functions of the variables.

NB. Other updates can be considered as long as they are monotonic. Extending guards require using a finer ordering.

Question. How does this compare to Minsky (counter) machines?

Some WSTS's: Monotonic Counter Machines



A run of M: $(\ell_0,0,1,4) \to (\ell_1,1,1,4) \to (\ell_2,1,0,4) \to (\ell_3,1,0,0)$

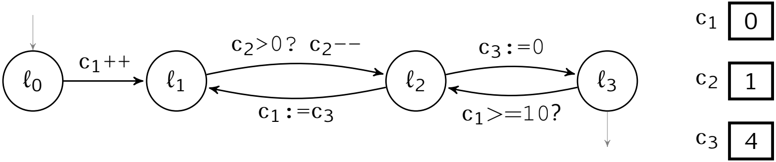
Ordering states: $(\ell_1,0,0,0) \leqslant (\ell_1,0,1,2)$ but $(\ell_1,0,0,0) \not\leqslant (\ell_2,0,1,2)$.
This is wqo as a product of wqo's: $(Loc,=) \times (\mathbb{N}^3,\leqslant_\times)$

Compatibility: easily checked when guards are upward-closed and assignments are monotonic functions of the variables.

NB. Other updates can be considered as long as they are monotonic. Extending guards require using a finer ordering.

Question. How does this compare to Minsky (counter) machines?

SOME WSTS'S: MONOTONIC COUNTER MACHINES



A run of M: $(\ell_0,0,1,4) \to (\ell_1,1,1,4) \to (\ell_2,1,0,4) \to (\ell_3,1,0,0)$

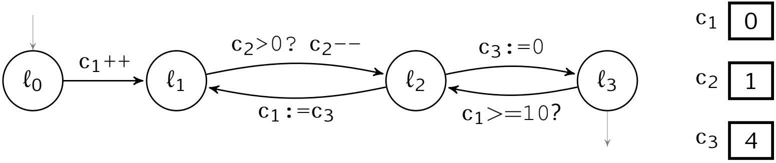
Ordering states: $(\ell_1,0,0,0) \leqslant (\ell_1,0,1,2)$ but $(\ell_1,0,0,0) \not\leqslant (\ell_2,0,1,2)$.
This is wqo as a product of wqo's: $(Loc,=) \times (\mathbb{N}^3,\leqslant_\times)$

Compatibility: easily checked when guards are upward-closed and assignments are monotonic functions of the variables.

NB. Other updates can be considered as long as they are monotonic. Extending guards require using a finer ordering.

Question. How does this compare to Minsky (counter) machines?

SOME WSTS'S: MONOTONIC COUNTER MACHINES



A run of M: $(\ell_0,0,1,4) \to (\ell_1,1,1,4) \to (\ell_2,1,0,4) \to (\ell_3,1,0,0)$

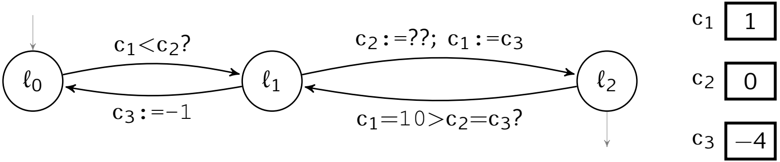
Ordering states: $(\ell_1,0,0,0) \leqslant (\ell_1,0,1,2)$ but $(\ell_1,0,0,0) \not\leqslant (\ell_2,0,1,2)$.
This is wqo as a product of wqo's: $(Loc,=) \times (\mathbb{N}^3,\leqslant_\times)$

Compatibility: easily checked when guards are upward-closed and assignments are monotonic functions of the variables.

NB. Other updates can be considered as long as they are monotonic. Extending guards require using a finer ordering.

Question. How does this compare to Minsky (counter) machines?

SOME WSTS'S: RELATIONAL AUTOMATA



Guards: comparisons between counters and constants

Updates: assignments with counter values and constants

One does not use $\leqslant_\times$ to compare states!! Rather

$$(a_1,\ldots,a_k) \leqslant_{\text{sparse}} (b_1,\ldots,b_k)$$

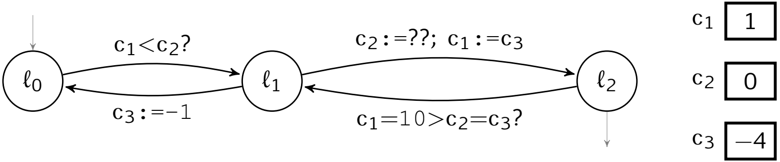
$$\overset{\text{def}}{\Leftrightarrow} \forall i,j=1,\ldots,k : \big(a_i \leqslant a_j \text{ iff } b_i \leqslant b_j\big) \land \big(|a_i - a_j| \leqslant |b_i - b_j|\big)\,.$$

Fact. $(\mathbb{Z}^k, \leqslant_{\text{sparse}})$ is wqo

Compatibility: We use $(\ell, a_1,\ldots,a_k) \leqslant (\ell', b_1,\ldots,b_k) \overset{\text{def}}{\Leftrightarrow}$

$$\ell = \ell' \land (a_1,\ldots,a_k,-1,10) \leqslant_{\text{sparse}} (b_1,\ldots,b_k,-1,10)\,.$$

SOME WSTS'S: RELATIONAL AUTOMATA



Guards: comparisons between counters and constants
Updates: assignments with counter values and constants

One does not use $\leqslant_\times$ to compare states!! Rather

$$(a_1,\ldots,a_k)\leqslant_{\text{sparse}}(b_1,\ldots,b_k)$$

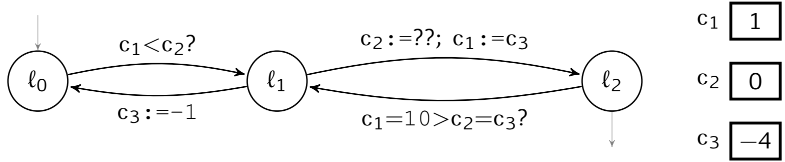
$$\overset{\text{def}}{\Leftrightarrow} \forall i,j = 1,\ldots,k : \big(a_i \leqslant a_j \text{ iff } b_i \leqslant b_j\big) \wedge \big(|a_i - a_j| \leqslant |b_i - b_j|\big) .$$

Fact. $(\mathbb{Z}^k, \leqslant_{\text{sparse}})$ is wqo

Compatibility: We use $(\ell, a_1,\ldots,a_k) \leqslant (\ell', b_1,\ldots,b_k) \overset{\text{def}}{\Leftrightarrow}$

$$\ell = \ell' \wedge (a_1,\ldots,a_k,-1,10) \leqslant_{\text{sparse}} (b_1,\ldots,b_k,-1,10) .$$

Some WSTS's: Relational automata



Guards: comparisons between counters and constants

Updates: assignments with counter values and constants

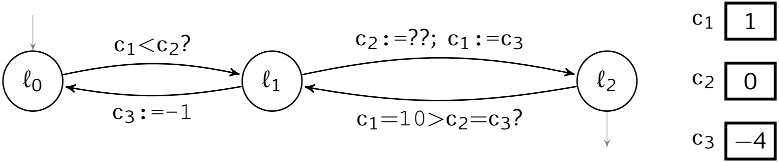
One does not use $\leqslant_\times$ to compare states!! Rather

$$(a_1,\ldots,a_k) \leqslant_{\text{sparse}} (b_1,\ldots,b_k)$$
$$\overset{\text{def}}{\Leftrightarrow} \forall i,j = 1,\ldots,k : \big(a_i \leqslant a_j \text{ iff } b_i \leqslant b_j\big) \wedge \big(|a_i - a_j| \leqslant |b_i - b_j|\big)\,.$$

Fact. $(\mathbb{Z}^k, \leqslant_{\text{sparse}})$ is wqo

Compatibility: We use $(\ell, a_1,\ldots,a_k) \leqslant (\ell', b_1,\ldots,b_k) \overset{\text{def}}{\Leftrightarrow}$
$$\ell = \ell' \wedge (a_1,\ldots,a_k,-1,10) \leqslant_{\text{sparse}} (b_1,\ldots,b_k,-1,10)\,.$$

Some WSTS's: Relational automata



Guards: comparisons between counters and constants
Updates: assignments with counter values and constants

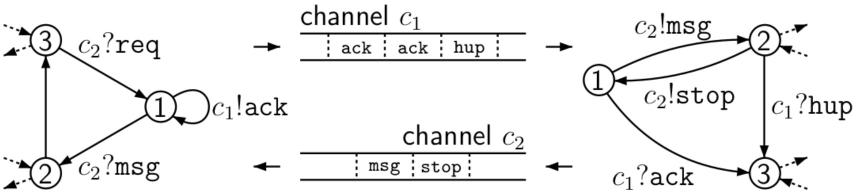
One does not use $\leqslant_\times$ to compare states!! Rather

$$(a_1,\ldots,a_k) \leqslant_{\textsf{sparse}} (b_1,\ldots,b_k)$$
$$\overset{\text{def}}{\Leftrightarrow} \forall i,j = 1,\ldots,k : \big(a_i \leqslant a_j \text{ iff } b_i \leqslant b_j\big) \wedge \big(|a_i - a_j| \leqslant |b_i - b_j|\big)\,.$$

Fact. $(\mathbb{Z}^k, \leqslant_{\textsf{sparse}})$ is wqo

Compatibility: We use $(\ell, a_1,\ldots,a_k) \leqslant (\ell', b_1,\ldots,b_k) \overset{\text{def}}{\Leftrightarrow}$
$$\ell = \ell' \wedge (a_1,\ldots,a_k,-1,10) \leqslant_{\textsf{sparse}} (b_1,\ldots,b_k,-1,10)\,.$$

Some WSTS's: LCS / Lossy Channel Systems



A configuration $\sigma = (\ell_1, \ell_2, w_1, w_2)$ with $w_i \in \Sigma^*$.
E.g., $w_1 = \texttt{hup.ack.ack}$.

Reliable steps: $\sigma \rightarrow_{\text{rel}} \rho$ read in front of channels, write at end (FIFO)

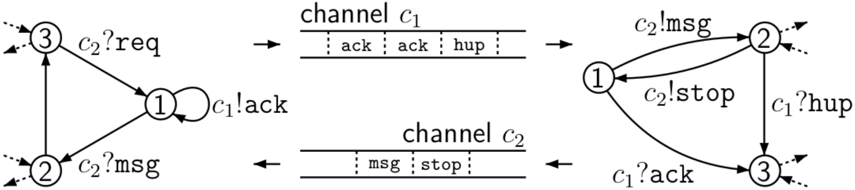
Lossy steps: messages may be lost nondeterministically

$$\sigma \rightarrow \sigma' \overset{\text{def}}{\Leftrightarrow} \sigma \sqsupseteq \rho \rightarrow_{\text{rel}} \rho' \sqsupseteq \sigma' \text{ for some } \rho, \rho'$$

where $(S, \sqsubseteq)$ is the wqo $(Loc_1, =) \times (Loc_2, =) \times (\Sigma^*, \leqslant_*)^{\{c_1, c_2\}}$

A model useful for concurrent protocols but also timed automata, metric temporal logic, products of modal logics, ...

Some WSTS's: LCS / Lossy Channel Systems



A configuration $\sigma = (\ell_1, \ell_2, w_1, w_2)$ with $w_i \in \Sigma^*$.
E.g., $w_1 = \texttt{hup.ack.ack}$.

Reliable steps: $\sigma \to_{\text{rel}} \rho$ read in front of channels, write at end (FIFO)

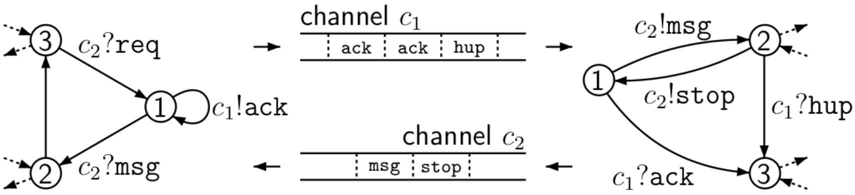
Lossy steps: messages may be lost nondeterministically

$$\sigma \to \sigma' \overset{\text{def}}{\Leftrightarrow} \sigma \sqsupseteq \rho \to_{\text{rel}} \rho' \sqsupseteq \sigma' \text{ for some } \rho, \rho'$$

where $(S, \sqsubseteq)$ is the wqo $(Loc_1, =) \times (Loc_2, =) \times (\Sigma^*, \leqslant_*)^{\{c_1, c_2\}}$

A model useful for concurrent protocols but also timed automata, metric temporal logic, products of modal logics, ...

SOME WSTS'S: LCS / LOSSY CHANNEL SYSTEMS



A configuration $\sigma = (\ell_1, \ell_2, w_1, w_2)$ with $w_i \in \Sigma^*$.
E.g., $w_1 = \texttt{hup.ack.ack}$.

Reliable steps: $\sigma \rightarrow_{\text{rel}} \rho$ read in front of channels, write at end (FIFO)

Lossy steps: messages may be lost nondeterministically

$$\sigma \rightarrow \sigma' \overset{\text{def}}{\Leftrightarrow} \sigma \sqsupseteq \rho \rightarrow_{\text{rel}} \rho' \sqsupseteq \sigma' \text{ for some } \rho, \rho'$$

where $(S, \sqsubseteq)$ is the wqo $(Loc_1, =) \times (Loc_2, =) \times (\Sigma^*, \leqslant_*)^{\{c_1, c_2\}}$

A model useful for concurrent protocols but also timed automata, metric temporal logic, products of modal logics, ...

WSTS VERIFICATION: TERMINATION

Def. A system terminates $\overset{\text{def}}{\Leftrightarrow}$ there are no infinite runs (starting from some given s_0)

Thm. "With minimal effectivity assumptions", termination is decidable for WSTS's

Indeed, if a WSTS has an infinite run, the infinite run contains an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ (by wqo)

But reciprocally, a finite run containing an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ can be extended to an infinite run (by compatibility), hence is a finite witness for non-termination!

Hence w.m.e.a. non-termination is r.e., i.e., termination is co-r.e.

Since w.m.e.a. termination is also r.e. (for systems with an image-finite transition relation), it is decidable.

Problem. Evaluate the complexity of this algorithm

WSTS verification: Termination

Def. A system terminates $\overset{\text{def}}{\Leftrightarrow}$ there are no infinite runs (starting from some given s_0)

Thm. "With minimal effectivity assumptions", termination is decidable for WSTS's

Indeed, if a WSTS has an infinite run, the infinite run contains an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ (by wqo)

But reciprocally, a finite run containing an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ can be extended to an infinite run (by compatibility), hence is a finite witness for non-termination!

Hence w.m.e.a. non-termination is r.e., i.e., termination is co-r.e.

Since w.m.e.a. termination is also r.e. (for systems with an image-finite transition relation), it is decidable.

Problem. Evaluate the complexity of this algorithm

WSTS VERIFICATION: TERMINATION

Def. A system terminates $\overset{\text{def}}{\Leftrightarrow}$ there are no infinite runs (starting from some given s_0)

Thm. “With minimal effectivity assumptions”, termination is decidable for WSTS’s

Indeed, if a WSTS has an infinite run, the infinite run contains an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ (by wqo)

But reciprocally, a finite run containing an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ can be extended to an infinite run (by compatibility), hence is a finite witness for non-termination!

Hence w.m.e.a. non-termination is r.e., i.e., termination is co-r.e.

Since w.m.e.a. termination is also r.e. (for systems with an image-finite transition relation), it is decidable.

Problem. Evaluate the complexity of this algorithm

WSTS verification: Termination

Def. A system terminates $\overset{\text{def}}{\Leftrightarrow}$ there are no infinite runs (starting from some given s_0)

Thm. “With minimal effectivity assumptions”, termination is decidable for WSTS’s

Indeed, if a WSTS has an infinite run, the infinite run contains an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ (by wqo)

But reciprocally, a finite run containing an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ can be extended to an infinite run (by compatibility), hence is a finite witness for non-termination!

Hence w.m.e.a. non-termination is r.e., i.e., termination is co-r.e.

Since w.m.e.a. termination is also r.e. (for systems with an image-finite transition relation), it is decidable.

Problem. Evaluate the complexity of this algorithm

WSTS verification: Termination

Def. A system terminates $\stackrel{\text{def}}{\Leftrightarrow}$ there are no infinite runs (starting from some given s_0)

Thm. "With minimal effectivity assumptions", termination is decidable for WSTS's

Indeed, if a WSTS has an infinite run, the infinite run contains an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ (by wqo)

But reciprocally, a finite run containing an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ can be extended to an infinite run (by compatibility), hence is a finite witness for non-termination!

Hence w.m.e.a. non-termination is r.e., i.e., termination is co-r.e.

Since w.m.e.a. termination is also r.e. (for systems with an image-finite transition relation), it is decidable.

Problem. Evaluate the complexity of this algorithm

WSTS verification: Termination

Def. A system terminates $\stackrel{\text{def}}{\Leftrightarrow}$ there are no infinite runs (starting from some given s_0)

Thm. "With minimal effectivity assumptions", termination is decidable for WSTS's

Indeed, if a WSTS has an infinite run, the infinite run contains an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ (by wqo)

But reciprocally, a finite run containing an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ can be extended to an infinite run (by compatibility), hence is a finite witness for non-termination!

Hence w.m.e.a. non-termination is r.e., i.e., termination is co-r.e.

Since w.m.e.a. termination is also r.e. (for systems with an image-finite transition relation), it is decidable.

Problem. Evaluate the complexity of this algorithm

WSTS VERIFICATION: TERMINATION

Def. A system terminates $\overset{\text{def}}{\Leftrightarrow}$ there are no infinite runs (starting from some given s_0)

Thm. "With minimal effectivity assumptions", termination is decidable for WSTS's

Indeed, if a WSTS has an infinite run, the infinite run contains an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ (by wqo)

But reciprocally, a finite run containing an increasing pair $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j \geqslant s_i$ can be extended to an infinite run (by compatibility), hence is a finite witness for non-termination!

Hence w.m.e.a. non-termination is r.e., i.e., termination is co-r.e.

Since w.m.e.a. termination is also r.e. (for systems with an image-finite transition relation), it is decidable.

Problem. Evaluate the complexity of this algorithm

WSTS Verification: Safety

Consider a set $B \subseteq S$ of "bad" states that is upward-closed.
E.g., a given error location, or a given location and some erroneous message.

Def. s_0 is safe in $S \stackrel{\text{def}}{\Leftrightarrow}$ no runs issued from s_0 ever visit B

Fact. $Pre^*(B) = \{s \in S \mid \exists t \in B \text{ with } s \xrightarrow{*} t\}$, the "unsafe states", is upward-closed (by compatibility)

Furthermore, $Pre^*(B)$ can be computed as the limit of
$B \subseteq Pre^{\leqslant 1}(B) \subseteq Pre^{\leqslant 2}(B) \subseteq \cdots \subseteq \bigcup_m Pre^{\leqslant m}(B) = Pre^*(B)$
(NB: $Pre^{\leqslant i}(B)$ too is upward-closed)

But a strictly increasing sequence of upward-closed subsets of a wqo is finite (recall: $(\mathcal{P}(X), \sqsubseteq_S)$ is well-founded iff X is wqo)

Cor. W.m.e.a. safety is decidable for WSTS's (& definable by excluded minors)

Problem. Evaluate the complexity of this algorithm

WSTS VERIFICATION: SAFETY

Consider a set $B \subseteq S$ of "bad" states that is upward-closed.
E.g., a given error location, or a given location and some erroneous message.

Def. s_0 is safe in $S \overset{\text{def}}{\Leftrightarrow}$ no runs issued from s_0 ever visit B

Fact. $Pre^*(B) = \{s \in S \mid \exists t \in B \text{ with } s \xrightarrow{*} t\}$, the "unsafe states", is upward-closed (by compatibility)

Furthermore, $Pre^*(B)$ can be computed as the limit of
$B \subseteq Pre^{\leqslant 1}(B) \subseteq Pre^{\leqslant 2}(B) \subseteq \cdots \subseteq \bigcup_m Pre^{\leqslant m}(B) = Pre^*(B)$
(NB: $Pre^{\leqslant i}(B)$ too is upward-closed)

But a strictly increasing sequence of upward-closed subsets of a wqo is finite (recall: $(\mathcal{P}(X), \sqsubseteq_S)$ is well-founded iff X is wqo)

Cor. W.m.e.a. safety is decidable for WSTS's (& definable by excluded minors)

Problem. Evaluate the complexity of this algorithm

WSTS Verification: Safety

Consider a set $B \subseteq S$ of "bad" states that is upward-closed.
E.g., a given error location, or a given location and some erroneous message.

Def. s_0 is safe in $S \overset{\text{def}}{\Leftrightarrow}$ no runs issued from s_0 ever visit B

Fact. $Pre^*(B) = \{s \in S \mid \exists t \in B \text{ with } s \xrightarrow{*} t\}$, the "unsafe states", is upward-closed (by compatibility)

Furthermore, $Pre^*(B)$ can be computed as the limit of
$B \subseteq Pre^{\leqslant 1}(B) \subseteq Pre^{\leqslant 2}(B) \subseteq \cdots \subseteq \bigcup_m Pre^{\leqslant m}(B) = Pre^*(B)$
(NB: $Pre^{\leqslant i}(B)$ too is upward-closed)

But a strictly increasing sequence of upward-closed subsets of a wqo is finite (recall: $(\mathcal{P}(X), \sqsubseteq_S)$ is well-founded iff X is wqo)

Cor. W.m.e.a. safety is decidable for WSTS's (& definable by excluded minors)

Problem. Evaluate the complexity of this algorithm

WSTS VERIFICATION: SAFETY

Consider a set $B \subseteq S$ of "bad" states that is upward-closed.
E.g., a given error location, or a given location and some erroneous message.

Def. s_0 is safe in $S \overset{\text{def}}{\Leftrightarrow}$ no runs issued from s_0 ever visit B

Fact. $Pre^*(B) = \{s \in S \mid \exists t \in B \text{ with } s \xrightarrow{*} t\}$, the "unsafe states", is upward-closed (by compatibility)

Furthermore, $Pre^*(B)$ can be computed as the limit of
$B \subseteq Pre^{\leqslant 1}(B) \subseteq Pre^{\leqslant 2}(B) \subseteq \cdots \subseteq \bigcup_m Pre^{\leqslant m}(B) = Pre^*(B)$
(NB: $Pre^{\leqslant i}(B)$ too is upward-closed)

But a strictly increasing sequence of upward-closed subsets of a wqo is finite (recall: $(\mathcal{P}(X), \sqsubseteq_S)$ is well-founded iff X is wqo)

Cor. W.m.e.a. safety is decidable for WSTS's (& definable by excluded minors)

Problem. Evaluate the complexity of this algorithm

WSTS Verification: Safety

Consider a set $B \subseteq S$ of "bad" states that is upward-closed.
E.g., a given error location, or a given location and some erroneous message.

Def. s_0 is safe in $S \overset{\text{def}}{\Leftrightarrow}$ no runs issued from s_0 ever visit B

Fact. $Pre^*(B) = \{s \in S \mid \exists t \in B \text{ with } s \xrightarrow{*} t\}$, the "unsafe states", is upward-closed (by compatibility)

Furthermore, $Pre^*(B)$ can be computed as the limit of
$B \subseteq Pre^{\leqslant 1}(B) \subseteq Pre^{\leqslant 2}(B) \subseteq \cdots \subseteq \bigcup_m Pre^{\leqslant m}(B) = Pre^*(B)$
(NB: $Pre^{\leqslant i}(B)$ too is upward-closed)

But a strictly increasing sequence of upward-closed subsets of a wqo is finite (recall: $(\mathcal{P}(X), \sqsubseteq_S)$ is well-founded iff X is wqo)

Cor. W.m.e.a. safety is decidable for WSTS's (& definable by excluded minors)

Problem. Evaluate the complexity of this algorithm