Undecidability methods for residuated lattices

Nick Galatos
(joint work with Gavin St. John)
University of Denver

June, 2019

Outline

Structure of the talk

The quasiequational theory of residuated lattices corresponds to the deducibility of the substructural logic $\mathbf{FL}$ and it is undecidable.

Outline

Structure of the talk

The quasiequational theory of residuated lattices corresponds to the deducibility of the substructural logic $\mathbf{FL}$ and it is undecidable. The same holds for various subvarieties: for example for knotted extensions.

Structure of the talk

The quasiequational theory of residuated lattices corresponds to the deducibility of the substructural logic $\mathbf{FL}$ and it is undecidable. The same holds for various subvarieties: for example for knotted extensions.

In the case of commutative residuated lattices, the variety has undecidable quasiequational theory, but its knotted subvarieties are decidable.

Structure of the talk

The quasiequational theory of residuated lattices corresponds to the deducibility of the substructural logic $\mathbf{FL}$ and it is undecidable.
The same holds for various subvarieties: for example for knotted extensions.

In the case of commutative residuated lattices, the variety has undecidable quasiequational theory, but its knotted subvarieties are decidable.
No proper subvariety of CRL was known to be undecidable.

Outline

Structure of the talk

The quasiequational theory of residuated lattices corresponds to the deducibility of the substructural logic $\mathbf{FL}$ and it is undecidable.
The same holds for various subvarieties: for example for knotted extensions.

In the case of commutative residuated lattices, the variety has undecidable quasiequational theory, but its knotted subvarieties are decidable.
No proper subvariety of CRL was known to be undecidable.

We will provide infinitely many subvarieties with undecidable quasiequational theory.

Outline

Structure of the talk

The quasiequational theory of residuated lattices corresponds to the deducibility of the substructural logic $\mathbf{FL}$ and it is undecidable.
The same holds for various subvarieties: for example for knotted extensions.

In the case of commutative residuated lattices, the variety has undecidable quasiequational theory, but its knotted subvarieties are decidable.
No proper subvariety of CRL was known to be undecidable.

We will provide infinitely many subvarieties with undecidable quasiequational theory.

- Residuated lattices
- Counter machines and encoding of RL
- Branching counter machines and encoding of CRL
- Exponential versions

A *residuated lattice*, is an algebra $\mathbf{L} = (L, \wedge, \vee, \cdot, \backslash, /, 1)$ such that

- $(L, \wedge, \vee)$ is a lattice,
- $(L, \cdot, 1)$ is a monoid and
- for all $a, b, c \in L$,

$$ab \leq c \;\Leftrightarrow\; b \leq a\backslash c \;\Leftrightarrow\; a \leq c/b.$$

Residuated lattices

A *residuated lattice*, is an algebra $\mathbf{L} = (L, \wedge, \vee, \cdot, \backslash, /, 1)$ such that

- $(L, \wedge, \vee)$ is a lattice,
- $(L, \cdot, 1)$ is a monoid and
- for all $a, b, c \in L$,

$$ab \leq c \;\Leftrightarrow\; b \leq a\backslash c \;\Leftrightarrow\; a \leq c/b.$$

If $xy = x \wedge y$ then $\mathbf{L}$ is a *Brouwerian algebra* (Heyting algebra, if there is a bottom element);

A *residuated lattice*, is an algebra $\mathbf{L} = (L, \wedge, \vee, \cdot, \backslash, /, 1)$ such that

- $(L, \wedge, \vee)$ is a lattice,
- $(L, \cdot, 1)$ is a monoid and
- for all $a, b, c \in L$,

$$ab \leq c \;\Leftrightarrow\; b \leq a\backslash c \;\Leftrightarrow\; a \leq c/b.$$

If $xy = x \wedge y$ then $\mathbf{L}$ is a *Brouwerian algebra* (Heyting algebra, if there is a bottom element); if it is complemented we obtain a Boolean algebra.

A *residuated lattice*, is an algebra $\mathbf{L} = (L, \wedge, \vee, \cdot, \backslash, /, 1)$ such that

- $(L, \wedge, \vee)$ is a lattice,
- $(L, \cdot, 1)$ is a monoid and
- for all $a, b, c \in L$,

$$ab \leq c \;\Leftrightarrow\; b \leq a\backslash c \;\Leftrightarrow\; a \leq c/b.$$

If $xy = x \wedge y$ then $\mathbf{L}$ is a *Brouwerian algebra* (Heyting algebra, if there is a bottom element); if it is complemented we obtain a Boolean algebra. If multiplication is commutative, we write $x \to y$ for $x\backslash y = y/x$.

Residuated lattices

A *residuated lattice*, is an algebra $\mathbf{L} = (L, \wedge, \vee, \cdot, \backslash, /, 1)$ such that

- $(L, \wedge, \vee)$ is a lattice,
- $(L, \cdot, 1)$ is a monoid and
- for all $a, b, c \in L$,

$$ab \leq c \Leftrightarrow b \leq a\backslash c \Leftrightarrow a \leq c/b.$$

If $xy = x \wedge y$ then $\mathbf{L}$ is a *Brouwerian algebra* (Heyting algebra, if there is a bottom element); if it is complemented we obtain a Boolean algebra. If multiplication is commutative, we write $x \to y$ for $x\backslash y = y/x$.

In every residuated lattice multiplication disctributes over join, so the reduct $(L, \vee, \cdot, 1)$ is an idempotent semiring.

Residuated lattices

A *residuated lattice*, is an algebra $\mathbf{L} = (L, \wedge, \vee, \cdot, \backslash, /, 1)$ such that

- $(L, \wedge, \vee)$ is a lattice,
- $(L, \cdot, 1)$ is a monoid and
- for all $a, b, c \in L$,

$$ab \le c \;\Leftrightarrow\; b \le a\backslash c \;\Leftrightarrow\; a \le c/b.$$

If $xy = x \wedge y$ then $\mathbf{L}$ is a *Brouwerian algebra* (Heyting algebra, if there is a bottom element); if it is complemented we obtain a Boolean algebra. If multiplication is commutative, we write $x \to y$ for $x\backslash y = y/x$.

In every residuated lattice multiplication disctributes over join, so the reduct $(L, \vee, \cdot, 1)$ is an idempotent semiring.

In general the lattice reduct need not be distributive, as in the lattice of ideals of a ring. $I \wedge J = I \cap J$, $I \vee J = I + J$, and IJ contains finite sums of products ij, as usual.

Residuated lattices

A *residuated lattice*, is an algebra $\mathbf{L} = (L, \wedge, \vee, \cdot, \backslash, /, 1)$ such that

- $(L, \wedge, \vee)$ is a lattice,
- $(L, \cdot, 1)$ is a monoid and
- for all $a, b, c \in L$,

$$ab \leq c \;\Leftrightarrow\; b \leq a\backslash c \;\Leftrightarrow\; a \leq c/b.$$

If $xy = x \wedge y$ then $\mathbf{L}$ is a *Brouwerian algebra* (Heyting algebra, if there is a bottom element); if it is complemented we obtain a Boolean algebra. If multiplication is commutative, we write $x \to y$ for $x\backslash y = y/x$.

In every residuated lattice multiplication disctributes over join, so the reduct $(L, \vee, \cdot, 1)$ is an idempotent semiring.

In general the lattice reduct need not be distributive, as in the lattice of ideals of a ring. $I \wedge J = I \cap J$, $I \vee J = I + J$, and IJ contains finite sums of products ij, as usual.

Other examples include lattice-ordered groups, relation algebras, models of logical systems (MV, BL, Relevance, linear).

Counter machines: hardware

Counter machines store numbers and can *increment*, *decrement* or *test* if the number is zero.

Counter machines: hardware

Counter machines store numbers and can *increment*, *decrement* or *test* if the number is zero.

More formally, the hardware of a *counter machine* consists of

- a finite set $R = \{r_1, \ldots, r_k\}$ of *registers*, which can be thought of as empty boxes labeled by the name of the register, and *tokens* each of which can be in some register,
- a final set Q of internal *states* in which the machine can be in, with designated initial state q_I and final state q_F.

Counter machines: hardware

Counter machines store numbers and can *increment*, *decrement* or *test* if the number is zero.

More formally, the hardware of a *counter machine* consists of

- a finite set $R = \{r_1, \ldots, r_k\}$ of *registers*, which can be thought of as empty boxes labeled by the name of the register, and *tokens* each of which can be in some register,
- a final set Q of internal *states* in which the machine can be in, with designated initial state q_I and final state q_F.

Therefore the configuration of a machine can be represented by the monoid term

$$qS_0 r_1^{n_1} S_1 \cdots S_{k-1} r_k^{n_k} S_k.$$

The auxiliary letters $S_0, \ldots, S_k$ are called *stoppers*.

Counter machines: software

Every machine always has the following instructions. For every letter x and every state q, we have the ambient instructions $xq \leq qx$ and $qx \leq xq$.

Counter machines: software

Every machine always has the following instructions. For every letter x and every state q, we have the ambient instructions $xq \leq qx$ and $qx \leq xq$.

The software consists of a finite set of instructions taken from three different types.

- Increment instructions: when in state q, increment register r_i by one token and change the internal state to q'. $qS_i \leq q'r_iS_i$

Counter machines: software

Every machine always has the following instructions. For every letter x and every state q, we have the ambient instructions $xq \leq qx$ and $qx \leq xq$.

The software consists of a finite set of instructions taken from three different types.

- Increment instructions: when in state q, increment register r_i by one token and change the internal state to q'. $qS_i \leq q' r_i S_i$
- Analogously we can decrement a non-zero counter (and do nothing if it is empty). $qr_i S_i \leq q' S_i$

Counter machines: software

Every machine always has the following instructions. For every letter x and every state q, we have the ambient instructions $xq \leq qx$ and $qx \leq xq$.

The software consists of a finite set of instructions taken from three different types.

- Increment instructions: when in state q, increment register r_i by one token and change the internal state to q'. $qS_i \leq q'r_iS_i$
- Analogously we can decrement a non-zero counter (and do nothing if it is empty). $qr_iS_i \leq q'S_i$
- Finally zero-test instructions: when in state q, check the contents of register r_i and if they are empty then move to state q'. $S_{i-1}qS_i \leq S_{i-1}q'S_i$.

Counter machines: software

Every machine always has the following instructions. For every letter x and every state q, we have the ambient instructions $xq \leq qx$ and $qx \leq xq$.

The software consists of a finite set of instructions taken from three different types.

- Increment instructions: when in state q, increment register r_i by one token and change the internal state to q'. $qS_i \leq q'r_iS_i$
- Analogously we can decrement a non-zero counter (and do nothing if it is empty). $qr_iS_i \leq q'S_i$
- Finally zero-test instructions: when in state q, check the contents of register r_i and if they are empty then move to state q'. $S_{i-1}qS_i \leq S_{i-1}q'S_i$.

The *computation relation* $\leq$ of a machine is defined as the reflexive-transitive closure of the smallest compatible relation containing the instructions.

Counter machines: software

Every machine always has the following instructions. For every letter x and every state q, we have the ambient instructions $xq \leq qx$ and $qx \leq xq$.

The software consists of a finite set of instructions taken from three different types.

- Increment instructions: when in state q, increment register r_i by one token and change the internal state to q'. $qS_i \leq q'r_iS_i$
- Analogously we can decrement a non-zero counter (and do nothing if it is empty). $qr_iS_i \leq q'S_i$
- Finally zero-test instructions: when in state q, check the contents of register r_i and if they are empty then move to state q'. $S_{i-1}qS_i \leq S_{i-1}q'S_i$.

The *computation relation* $\leq$ of a machine is defined as the reflexive-transitive closure of the smallest compatible relation containing the instructions.
We say that a configuration C is *accepted* if $C \leq q_F S_0 S_1 \cdots S_k$.

Counter machines: example

For example, consider the machine that has set of states $Q = \{q_1, q_F\}$, with initial and final state q_F, set of registers $R = \{r_1, r_2\}$ and set of instructions $P = \{q_F r_1 S_1 \leq q_1 S_1,\ q_1 r_2 S_2 \leq q_F S_2\}$, then we have

$$
\begin{aligned}
q_F S_0 r_1 S_1 r_2 S_2 &\leq S_0 q_F r_1 S_1 r_2 S_2 \\
&\leq S_0 q_1 S_1 r_2 S_2 \\
&\leq S_0 S_1 q_1 r_2 S_2 \\
&\leq S_0 S_1 q_F S_2 \\
&\leq S_0 q_F S_1 S_2 \\
&\leq q_F S_0 S_1 S_2.
\end{aligned}
$$

The only initial configurations that are accepted are of the form $q_F S_0 r_1^n S_1 r_2^n S_2$, where n is a natural number, so the machine checks if we have an equal number of r_1-tokens as r_2-tokens.

Counter machines undecidability

It is well known that there is a counter machine with an undecidable set of accepted configurations.

Counter machines undecidability

It is well known that there is a counter machine with an undecidable set of accepted configurations. All computations in this machine are interpreted as valid in the residuated lattice presented by relations corresponding to the instructions.

Counter machines undecidability

It is well known that there is a counter machine with an undecidable set of accepted configurations. All computations in this machine are interpreted as valid in the residuated lattice presented by relations corresponding to the instructions.

Therefore, RL satisfies the quasiequation

$$\& P \Rightarrow u \leq q_F$$

where P is the set of instructions of an undecidable machine and u is an accepted configuration of the machine.

Counter machines undecidability

It is well known that there is a counter machine with an undecidable set of accepted configurations. All computations in this machine are interpreted as valid in the residuated lattice presented by relations corresponding to the instructions.

Therefore, RL satisfies the quasiequation

$$\&P \;\Rightarrow\; u \leq q_F$$

where P is the set of instructions of an undecidable machine and u is an accepted configuration of the machine.

Conversely, if some configuration is not accepted then we can construct a residuated lattice that falsifies the quasiequation.

Counter machines undecidability

It is well known that there is a counter machine with an undecidable set of accepted configurations. All computations in this machine are interpreted as valid in the residuated lattice presented by relations corresponding to the instructions.

Therefore, RL satisfies the quasiequation

$$\& P \;\Rightarrow\; u \leq q_F$$

where P is the set of instructions of an undecidable machine and u is an accepted configuration of the machine.

Conversely, if some configuration is not accepted then we can construct a residuated lattice that falsifies the quasiequation.

Therefore, the word problem for residuated lattices is undecidable.

Counter machines undecidability

It is well known that there is a counter machine with an undecidable set of accepted configurations. All computations in this machine are interpreted as valid in the residuated lattice presented by relations corresponding to the instructions.

Therefore, RL satisfies the quasiequation

$$\&P \;\Rightarrow\; u \leq q_F$$

where P is the set of instructions of an undecidable machine and u is an accepted configuration of the machine.

Conversely, if some configuration is not accepted then we can construct a residuated lattice that falsifies the quasiequation.

Therefore, the word problem for residuated lattices is undecidable.

The counterexample is constructed using *residuated frames*.

A *residuated frame* is a structure $\mathbf{W} = (W, \circ, \varepsilon, N, W')$ where

- N is a binary relation between the sets W and W'
- $(W, \circ, \varepsilon)$ is a monoid
- for all $x, y \in W$ and $z \in W'$ there exist $x \mathbin{\backslash\!\backslash} z, z \mathbin{/\!/} y$ in W' such that (nuclearity)

$$(x \circ y) \; N \; z \;\; \Leftrightarrow \;\; y \; N \; (x \mathbin{\backslash\!\backslash} z) \;\; \Leftrightarrow \;\; x \; N \; (z \mathbin{/\!/} y).$$

A *residuated frame* is a structure $\mathbf{W} = (W, \circ, \varepsilon, N, W')$ where

- N is a binary relation between the sets W and W'
- $(W, \circ, \varepsilon)$ is a monoid
- for all $x, y \in W$ and $z \in W'$ there exist $x \mathbin{\backslash\!\backslash} z, z \mathbin{/\!/} y$ in W' such that (nuclearity)

$$(x \circ y)\ N\ z \iff y\ N\ (x \mathbin{\backslash\!\backslash} z) \iff x\ N\ (z \mathbin{/\!/} y).$$

For $X \subseteq W$ and $Y \subseteq W'$ we define

$$X^{\rhd} = \{b \in W' : x\ N\ b, \text{ for all } x \in X\}$$
$$Y^{\lhd} = \{a \in W : a\ N\ y, \text{ for all } y \in Y\}$$

A *residuated frame* is a structure $\mathbf{W} = (W, \circ, \varepsilon, N, W')$ where

- N is a binary relation between the sets W and W'
- $(W, \circ, \varepsilon)$ is a monoid
- for all $x, y \in W$ and $z \in W'$ there exist $x \mathbin{\backslash\!\!\backslash} z, z \mathbin{/\!\!/} y$ in W' such that (nuclearity)

$$(x \circ y) \; N \; z \iff y \; N \; (x \mathbin{\backslash\!\!\backslash} z) \iff x \; N \; (z \mathbin{/\!\!/} y).$$

For $X \subseteq W$ and $Y \subseteq W'$ we define

$$X^{\rhd} = \{b \in W' : x \; N \; b, \text{ for all } x \in X\}$$
$$Y^{\lhd} = \{a \in W : a \; N \; y, \text{ for all } y \in Y\}$$

We define $\gamma(X) = X^{\rhd\lhd}$.

A *residuated frame* is a structure $\mathbf{W} = (W, \circ, \varepsilon, N, W')$ where

- N is a binary relation between the sets W and W'
- $(W, \circ, \varepsilon)$ is a monoid
- for all $x, y \in W$ and $z \in W'$ there exist $x \mathbin{\backslash\!\!\backslash} z, z \mathbin{/\!\!/} y$ in W' such that (nuclearity)

$$(x \circ y) \; N \; z \quad \Leftrightarrow \quad y \; N \; (x \mathbin{\backslash\!\!\backslash} z) \quad \Leftrightarrow \quad x \; N \; (z \mathbin{/\!\!/} y).$$

For $X \subseteq W$ and $Y \subseteq W'$ we define

$$X^{\rhd} = \{b \in W' : x \; N \; b, \text{ for all } x \in X\}$$
$$Y^{\lhd} = \{a \in W : a \; N \; y, \text{ for all } y \in Y\}$$

We define $\gamma(X) = X^{\rhd\lhd}$.

Corollary. If $\mathbf{W}$ is a residuated frame then the *Galois/dual algebra* $\mathbf{W}^+ = (\gamma[\mathcal{P}(W)], \cap, \cup_\gamma, \circ_\gamma, \gamma(1), \backslash, /)$ is a residuated lattice, where
$X \circ Y = \{x \circ y : x \in X, y \in Y\}$,
$X \backslash Y = \{z : X \circ \{z\} \subseteq Y\}$

The residuated frame

Let M be a machine and $W := (Q \cup R_k \cup S)^*$ be the free monoid generated by $Q \cup R_k \cup S$ and $W' = W \times W$.

The residuated frame

Let M be a machine and $W := (Q \cup R_k \cup S)^*$ be the free monoid generated by $Q \cup R_k \cup S$ and $W' = W \times W$. We define the relation $N \subseteq W \times W'$ via

$$x \; N \; (u, v) \quad \text{iff} \quad uxv \leq q_F,$$

for all $x, z \in W$.

The residuated frame

Let M be a machine and $W := (Q \cup R_k \cup S)^*$ be the free monoid generated by $Q \cup R_k \cup S$ and $W' = W \times W$. We define the relation $N \subseteq W \times W'$ via

$$x \; N \; (u, v) \quad \text{iff} \quad uxv \leq q_F,$$

for all $x, z \in W$. Observe that, for any $x, y, u, v \in W$,

$$xy \; N \; (u, v) \iff uxyv \leq q_F \iff x \; N \; (u, yv) \iff y \; N \; (ux, v).$$

It follows that N is nuclear.

The residuated frame

Let M be a machine and $W := (Q \cup R_k \cup S)^*$ be the free monoid generated by $Q \cup R_k \cup S$ and $W' = W \times W$. We define the relation $N \subseteq W \times W'$ via

$$x \; N \; (u, v) \quad \text{iff} \quad uxv \leq q_F,$$

for all $x, z \in W$. Observe that, for any $x, y, u, v \in W$,

$$xy \; N \; (u, v) \iff uxyv \leq q_F \iff x \; N \; (u, yv) \iff y \; N \; (ux, v).$$

It follows that N is nuclear.

$\mathbf{W} := (W, W', N, \cdot, 1)$ is a residuated frame, $\mathbf{W}^+ \in$ RL, and there exists a valuation $\nu : \mathbf{Fm} \to \mathbf{W}^+$ that falsifies the quasiequation of the machine.

Resilience of the encoding

We consider the subvariety axiomatized by the equation $x^2y^2 = y^2x^2$.

Resilience of the encoding

We consider the subvariety axiomatized by the equation $x^2y^2 = y^2x^2$. Since the inequality $x^2y^2 \leq y^2x^2$ is valid in this subvariety, the computation relation needs to allow for such transitions.

Resilience of the encoding

We consider the subvariety axiomatized by the equation $x^2y^2 = y^2x^2$. Since the inequality $x^2y^2 \leq y^2x^2$ is valid in this subvariety, the computation relation needs to allow for such transitions. From the perspective of the computation, an application of this inequality (since it is not a specific instruction of the machine) may be viewed as a *glitch* or a *bug* in the computation which may be applied at any time without our control.

Resilience of the encoding

We consider the subvariety axiomatized by the equation $x^2y^2 = y^2x^2$. Since the inequality $x^2y^2 \leq y^2x^2$ is valid in this subvariety, the computation relation needs to allow for such transitions. From the perspective of the computation, an application of this inequality (since it is not a specific instruction of the machine) may be viewed as a *glitch* or a *bug* in the computation which may be applied at any time without our control. The obvious worry is then that the computation relation will be different and may not correspond any more to an undecidable machine.

Resilience of the encoding

We consider the subvariety axiomatized by the equation $x^2y^2 = y^2x^2$. Since the inequality $x^2y^2 \leq y^2x^2$ is valid in this subvariety, the computation relation needs to allow for such transitions. From the perspective of the computation, an application of this inequality (since it is not a specific instruction of the machine) may be viewed as a *glitch* or a *bug* in the computation which may be applied at any time without our control. The obvious worry is then that the computation relation will be different and may not correspond any more to an undecidable machine.
Fortunately, the same encoding and arguments work:

Resilience of the encoding

We consider the subvariety axiomatized by the equation $x^2y^2 = y^2x^2$. Since the inequality $x^2y^2 \leq y^2x^2$ is valid in this subvariety, the computation relation needs to allow for such transitions. From the perspective of the computation, an application of this inequality (since it is not a specific instruction of the machine) may be viewed as a *glitch* or a *bug* in the computation which may be applied at any time without our control. The obvious worry is then that the computation relation will be different and may not correspond any more to an undecidable machine.

Fortunately, the same encoding and arguments work: Even though instances of the inequality $x^2y^2 \leq y^2x^2$ are available, they cannot be applied to any configuration in a non-trivial way, due to the positioning of the stoppers in configurations.

Resilience of the encoding

We consider the subvariety axiomatized by the equation $x^2y^2 = y^2x^2$. Since the inequality $x^2y^2 \leq y^2x^2$ is valid in this subvariety, the computation relation needs to allow for such transitions. From the perspective of the computation, an application of this inequality (since it is not a specific instruction of the machine) may be viewed as a *glitch* or a *bug* in the computation which may be applied at any time without our control. The obvious worry is then that the computation relation will be different and may not correspond any more to an undecidable machine.

Fortunately, the same encoding and arguments work: Even though instances of the inequality $x^2y^2 \leq y^2x^2$ are available, they cannot be applied to any configuration in a non-trivial way, due to the positioning of the stoppers in configurations. Therefore, the word problem for the subvariety axiomatized by $x^2y^2 = y^2x^2$ is undecidable.

Resilience of the encoding

We consider the subvariety axiomatized by the equation $x^2y^2 = y^2x^2$. Since the inequality $x^2y^2 \leq y^2x^2$ is valid in this subvariety, the computation relation needs to allow for such transitions. From the perspective of the computation, an application of this inequality (since it is not a specific instruction of the machine) may be viewed as a *glitch* or a *bug* in the computation which may be applied at any time without our control. The obvious worry is then that the computation relation will be different and may not correspond any more to an undecidable machine.

Fortunately, the same encoding and arguments work: Even though instances of the inequality $x^2y^2 \leq y^2x^2$ are available, they cannot be applied to any configuration in a non-trivial way, due to the positioning of the stoppers in configurations. Therefore, the word problem for the subvariety axiomatized by $x^2y^2 = y^2x^2$ is undecidable.

Unfortunately, the encoding does not work for commutativity $xy = yx$, as by applications of commutativity tokens can move past the stoppers and then the zero-test may not be implemented correctly. A different encoding is needed.

Different encoding

The quasiequational theory in the $\{\cdot, 1, \leq\}$-fragment is actually decidable, so no encoding will work in this language.

Different encoding

The quasiequational theory in the $\{\cdot, 1, \leq\}$-fragment is actually decidable, so no encoding will work in this language. The idea is to involve the connective of join $\vee$ to implement *parallel computation*.

Different encoding

The quasiequational theory in the $\{\cdot, 1, \leq\}$-fragment is actually decidable, so no encoding will work in this language. The idea is to involve the connective of join $\vee$ to implement *parallel computation*. The main strand of the computation proceeds on transitioning to the next state by the zero-test without any restrictions, while the auxiliary computation safeguards that the zero-test is applied correctly, by terminating only when the value of the register was empty.

Different encoding

The quasiequational theory in the $\{\cdot, 1, \leq\}$-fragment is actually decidable, so no encoding will work in this language. The idea is to involve the connective of join $\vee$ to implement *parallel computation*. The main strand of the computation proceeds on transitioning to the next state by the zero-test without any restrictions, while the auxiliary computation safeguards that the zero-test is applied correctly, by terminating only when the value of the register was empty. Therefore, we consider joins of configurations, which we call *instantaneous descriptions*, IDs. and we represent by $C_1 \vee \cdots \vee C_m$, where the C_is are configurations; so IDs of the machine are elements of the free join-semilattice over the set QR^*. We assume that this sits inside a commutative idempotent semiring generated by $Q \cup R$.

Different encoding

The quasiequational theory in the $\{\cdot, 1, \leq\}$-fragment is actually decidable, so no encoding will work in this language. The idea is to involve the connective of join $\vee$ to implement *parallel computation*. The main strand of the computation proceeds on transitioning to the next state by the zero-test without any restrictions, while the auxiliary computation safeguards that the zero-test is applied correctly, by terminating only when the value of the register was empty. Therefore, we consider joins of configurations, which we call *instantaneous descriptions*, IDs. and we represent by $C_1 \vee \cdots \vee C_m$, where the C_is are configurations; so IDs of the machine are elements of the free join-semilattice over the set QR^*. We assume that this sits inside a commutative idempotent semiring generated by $Q \cup R$. Due to commutativity, stoppers play no role (and are omitted) and monoid words are of the form $qr_1^{n_1} r_2^{n_2} \cdots r_k^{n_k}$.

Different encoding

The quasiequational theory in the $\{\cdot, 1, \leq\}$-fragment is actually decidable, so no encoding will work in this language. The idea is to involve the connective of join $\vee$ to implement *parallel computation*. The main strand of the computation proceeds on transitioning to the next state by the zero-test without any restrictions, while the auxiliary computation safeguards that the zero-test is applied correctly, by terminating only when the value of the register was empty. Therefore, we consider joins of configurations, which we call *instantaneous descriptions*, IDs. and we represent by $C_1 \vee \cdots \vee C_m$, where the C_is are configurations; so IDs of the machine are elements of the free join-semilattice over the set QR^*. We assume that this sits inside a commutative idempotent semiring generated by $Q \cup R$. Due to commutativity, stoppers play no role (and are omitted) and monoid words are of the form $qr_1^{n_1} r_2^{n_2} \cdots r_k^{n_k}$.

We call this formalization an *And-branching Counter Machine*.

Computation relation

- $q \leq q'r$ **Add**: for state q adding a token r and state q'.

Computation relation

- $q \leq q'r$ **Add**: for state q adding a token r and state q'.
- $qr \leq q'$ **Substract**: for state q erasing an existing colored token r and state q'.

Computation relation

- $q \leq q'r$ **Add**: for state q adding a token r and state q'.
- $qr \leq q'$ **Substract**: for state q erasing an existing colored token r and state q'.
- $q \leq q' \vee q''$ **Copy**: for state q creating a copy of the data and for each copy moving to states q' and q''.

Computation relation

- $q \leq q'r$ **Add**: for state q adding a token r and state q'.
- $qr \leq q'$ **Substract**: for state q erasing an existing colored token r and state q'.
- $q \leq q' \vee q''$ **Copy**: for state q creating a copy of the data and for each copy moving to states q' and q''.

We say that the machine *terminates* on a given ID u, if $u \leq q_F \vee \cdots \vee q_F$.

Examples, the even machine

Let M_{even} be a machine with only one counter r three states q_0, q_1, q_f (where $q_I = q_0$), and three instructions which we even name for reference as p_1, p_2, p_3:

$$q_0 r \leq^{p_1} q_1 \quad q_1 r \leq^{p_2} q_0 \quad q_0 \leq^{p_3} q_F$$

Examples, the even machine

Let M_{even} be a machine with only one counter r three states q_0, q_1, q_f (where $q_I = q_0$), and three instructions which we even name for reference as p_1, p_2, p_3:

$$q_0 r \leq^{p_1} q_1 \quad q_1 r \leq^{p_2} q_0 \quad q_0 \leq^{p_3} q_F$$

Note that $q_0 r^n \leq q_F$ iff n is even. For example:

Examples, the even machine

Let M_{even} be a machine with only one counter r three states q_0, q_1, q_f (where $q_I = q_0$), and three instructions which we even name for reference as p_1, p_2, p_3:

$$q_0 r \leq^{p_1} q_1 \quad q_1 r \leq^{p_2} q_0 \quad q_0 \leq^{p_3} q_F$$

Note that $q_0 r^n \leq q_F$ iff n is even. For example:

$$q_0 r^4 \leq^{p_1} q_1 r^3 \leq^{p_2} q_0 r^2 \leq^{p_1} q_1 r \leq^{p_2} q_0 \leq^{p_3} q_F$$

Examples, the even machine

Let M_{even} be a machine with only one counter r three states q_0, q_1, q_f (where $q_I = q_0$), and three instructions which we even name for reference as p_1, p_2, p_3:

$$q_0 r \leq^{p_1} q_1 \quad q_1 r \leq^{p_2} q_0 \quad q_0 \leq^{p_3} q_F$$

Note that $q_0 r^n \leq q_F$ iff n is even. For example:

$$q_0 r^4 \leq^{p_1} q_1 r^3 \leq^{p_2} q_0 r^2 \leq^{p_1} q_1 r \leq^{p_2} q_0 \leq^{p_3} q_F$$

$$q_0 r^3 \leq^{p_1} q_1 r^2 \leq^{p_2} q_0 r \leq^{p_3} q_F r$$

Examples, the zero test

Zero-test (for r_3): Let $R = \{r_1, r_2, r_3\}$, $Q = \{q_I, q', z_3, q_F\}$ and assume that the instructions include

$$q_I \leq q' \vee z_3 \quad z_3 r_1 \leq z_3 \quad z_3 r_2 \leq z_3 \quad z_3 \leq q_F$$

and possibly other instructions of the form $q' \leq \ldots$.

Examples, the zero test

Zero-test (for r_3): Let $R = \{r_1, r_2, r_3\}$, $Q = \{q_I, q', z_3, q_F\}$ and assume that the instructions include

$$q_I \leq q' \vee z_3 \quad z_3 r_1 \leq z_3 \quad z_3 r_2 \leq z_3 \quad z_3 \leq q_F$$

and possibly other instructions of the form $q' \leq \ldots$.

Notice that

$q_I r_1^2 r_2 r_3 \leq (q' \vee z_3) r_1^2 r_2 r_3 = q' r_1^2 r_2 r_3 \vee z_3 r_1^2 r_2 r_3 \leq \ldots q' r_1^2 r_2 r_3 \vee q_F r_3$

Since $q_F r_3 \not\leq q_F$ the process does not terminate.

Examples, the zero test

Zero-test (for r_3): Let $R = \{r_1, r_2, r_3\}$, $Q = \{q_I, q', z_3, q_F\}$ and assume that the instructions include

$$q_I \leq q' \vee z_3 \qquad z_3 r_1 \leq z_3 \qquad z_3 r_2 \leq z_3 \qquad z_3 \leq q_F$$

and possibly other instructions of the form $q' \leq \ldots$.

Notice that
$q_I r_1^2 r_2 r_3 \leq (q' \vee z_3) r_1^2 r_2 r_3 = q' r_1^2 r_2 r_3 \vee z_3 r_1^2 r_2 r_3 \leq \ldots q' r_1^2 r_2 r_3 \vee q_F r_3$
Since $q_F r_3 \not\leq q_F$ the process does not terminate.

However,
$q_I r_1^2 r_2 \leq (q' \vee z_3) r_1^2 r_2 = q' r_1^2 r_2 \vee z_3 r_1^2 r_2 \leq \ldots \leq q_F \vee q_F = q_F$ if we assume that due to other instructions we get $q' r_1^2 r_2 \leq q_F$.

Examples, the zero test

Zero-test (for r_3): Let $R = \{r_1, r_2, r_3\}$, $Q = \{q_I, q', z_3, q_F\}$ and assume that the instructions include

$$q_I \leq q' \vee z_3 \quad z_3 r_1 \leq z_3 \quad z_3 r_2 \leq z_3 \quad z_3 \leq q_F$$

and possibly other instructions of the form $q' \leq \ldots$.

Notice that
$q_I r_1^2 r_2 r_3 \leq (q' \vee z_3) r_1^2 r_2 r_3 = q' r_1^2 r_2 r_3 \vee z_3 r_1^2 r_2 r_3 \leq \ldots q' r_1^2 r_2 r_3 \vee q_F r_3$
Since $q_F r_3 \not\leq q_F$ the process does not terminate.

However,
$q_I r_1^2 r_2 \leq (q' \vee z_3) r_1^2 r_2 = q' r_1^2 r_2 \vee z_3 r_1^2 r_2 \leq \ldots \leq q_F \vee q_F = q_F$ if we assume that due to other instructions we get $q' r_1^2 r_2 \leq q_F$.

So, these instructions block termination if there are r_3 tokens, but allow the continuation of the passage to q' (in the context of a non-harmful $\_ \vee q_F$) if the contents of r_3 are empty.

Examples, the zero test

Zero-test (for r_3): Let $R = \{r_1, r_2, r_3\}$, $Q = \{q_I, q', z_3, q_F\}$ and assume that the instructions include

$$q_I \leq q' \vee z_3 \quad z_3 r_1 \leq z_3 \quad z_3 r_2 \leq z_3 \quad z_3 \leq q_F$$

and possibly other instructions of the form $q' \leq \ldots$.

Notice that
$q_I r_1^2 r_2 r_3 \leq (q' \vee z_3) r_1^2 r_2 r_3 = q' r_1^2 r_2 r_3 \vee z_3 r_1^2 r_2 r_3 \leq \ldots q' r_1^2 r_2 r_3 \vee q_F r_3$
Since $q_F r_3 \not\leq q_F$ the process does not terminate.

However,
$q_I r_1^2 r_2 \leq (q' \vee z_3) r_1^2 r_2 = q' r_1^2 r_2 \vee z_3 r_1^2 r_2 \leq \ldots \leq q_F \vee q_F = q_F$ if we assume that due to other instructions we get $q' r_1^2 r_2 \leq q_F$.

So, these instructions block termination if there are r_3 tokens, but allow the continuation of the passage to q' (in the context of a non-harmful $\_ \vee q_F$) if the contents of r_3 are empty. (In this case if $q'x \leq q_F$ then $qx \leq q_F$, *for all* x *that do not contain* r_3 and we can denote this fact by $qx \sqsubseteq q'x$.)

Examples, the zero test

Zero-test (for r_3): Let $R = \{r_1, r_2, r_3\}$, $Q = \{q_I, q', z_3, q_F\}$ and assume that the instructions include

$$q_I \leq q' \vee z_3 \quad z_3 r_1 \leq z_3 \quad z_3 r_2 \leq z_3 \quad z_3 \leq q_F$$

and possibly other instructions of the form $q' \leq \dots$.

Notice that
$q_I r_1^2 r_2 r_3 \leq (q' \vee z_3) r_1^2 r_2 r_3 = q' r_1^2 r_2 r_3 \vee z_3 r_1^2 r_2 r_3 \leq \dots q' r_1^2 r_2 r_3 \vee q_F r_3$
Since $q_F r_3 \not\leq q_F$ the process does not terminate.

However,
$q_I r_1^2 r_2 \leq (q' \vee z_3) r_1^2 r_2 = q' r_1^2 r_2 \vee z_3 r_1^2 r_2 \leq \dots \leq q_F \vee q_F = q_F$ if we assume that due to other instructions we get $q' r_1^2 r_2 \leq q_F$.

So, these instructions block termination if there are r_3 tokens, but allow the continuation of the passage to q' (in the context of a non-harmful $\_ \vee q_F$) if the contents of r_3 are empty. (In this case if $q' x \leq q_F$ then $qx \leq q_F$, *for all* x *that do not contain* r_3 and we can denote this fact by $qx \sqsubseteq q' x$.)

Theorem [Lincoln et al., 1992] The quasiequational theory of CRL is undecidabile.

Known results

As a side note, the equational theory is decidable. (Via proof theory using a Gentzen-style calculus and proving cut-elimination. Alternatively by using Residuated Frames [G.-Jipsen])

Known results

As a side note, the equational theory is decidable. (Via proof theory using a Gentzen-style calculus and proving cut-elimination. Alternatively by using Residuated Frames [G.-Jipsen])

Theorem. [Chvalovsky-Horcik] RL $+ (x^n \leq x^m)$ has an undecidable word problem (hence also quasiequational theory), for most n, m. (By using machines and also encoding square-free words.)

Known results

As a side note, the equational theory is decidable. (Via proof theory using a Gentzen-style calculus and proving cut-elimination. Alternatively by using Residuated Frames [G.-Jipsen])

Theorem. [Chvalovsky-Horcik] RL $+ (x^n \leq x^m)$ has an undecidable word problem (hence also quasiequational theory), for most n, m. (By using machines and also encoding square-free words.)

We would like to prove the undecidability of the quasiequational theory for subvarieties of CRL, for example CRL $+(x \leq x^2)$.

Known results

As a side note, the equational theory is decidable. (Via proof theory using a Gentzen-style calculus and proving cut-elimination. Alternatively by using Residuated Frames [G.-Jipsen])

Theorem. [Chvalovsky-Horcik] RL $+ (x^n \leq x^m)$ has an undecidable word problem (hence also quasiequational theory), for most n, m. (By using machines and also encoding square-free words.)

We would like to prove the undecidability of the quasiequational theory for subvarieties of CRL, for example CRL $+(x \leq x^2)$.

The same encoding fails: The $\leq$ transitioning relation of the machine needs to satisfy $x \leq x^2$. So, in the middle of the computation such inequalities/bugs/glitches may be applied and contents of registers can increase spontaneously.

Known results

As a side note, the equational theory is decidable. (Via proof theory using a Gentzen-style calculus and proving cut-elimination. Alternatively by using Residuated Frames [G.-Jipsen])

Theorem. [Chvalovsky-Horcik] RL + $(x^n \leq x^m)$ has an undecidable word problem (hence also quasiequational theory), for most n, m. (By using machines and also encoding square-free words.)

We would like to prove the undecidability of the quasiequational theory for subvarieties of CRL, for example CRL $+(x \leq x^2)$.

The same encoding fails: The $\leq$ transitioning relation of the machine needs to satisfy $x \leq x^2$. So, in the middle of the computation such inequalities/bugs/glitches may be applied and contents of registers can increase spontaneously.
For example this interferes with the computation of the machine M_{even} which then accepts all ID's.

This is not merely a problem with the encoding:

Other subvarieties

This is not merely a problem with the encoding:

Theorem. [Kripke] The equational theory of CRL+$(x \leq x^2)$

Other subvarieties

This is not merely a problem with the encoding:

Theorem. [Kripke] The equational theory of CRL$+(x \leq x^2)$ and [Urquhart] the quasiequational theory are decidable.

Other subvarieties

This is not merely a problem with the encoding:

Theorem. [Kripke] The equational theory of CRL+$(x \leq x^2)$ and [Urquhart] the quasiequational theory are decidable.

Theorem. [Blok, van Alten] CRL + $(x^n \leq x^m)$ has the Finite Embeddability Property (for practically all n, m); hence decidable quasiequational theory.

Equations

In an equation over $\{\cdot, 1, \vee\}$ we can distribute products over joins,

Equations

In an equation over $\{\cdot, 1, \vee\}$ we can distribute products over joins, then break it in two inequalities each of the form $s_1 \vee \cdots \vee s_n \leq t_1 \vee \cdots \vee t_m$,

Equations

In an equation over $\{\cdot, 1, \vee\}$ we can distribute products over joins, then break it in two inequalities each of the form $s_1 \vee \cdots \vee s_n \leq t_1 \vee \cdots \vee t_m$, and unltimately into a conjunction of equations of the form $s \leq t_1 \vee \cdots \vee t_m$, where s, t_j: monoid terms.

Equations

In an equation over $\{\cdot, 1, \vee\}$ we can distribute products over joins, then break it in two inequalities each of the form $s_1 \vee \cdots \vee s_n \leq t_1 \vee \cdots \vee t_m$, and unltimately into a conjunction of equations of the form $s \leq t_1 \vee \cdots \vee t_m$, where s, t_j: monoid terms.

We proceed by example: $x^2 y \leq xy \vee yx$

Equations

In an equation over $\{\cdot, 1, \vee\}$ we can distribute products over joins, then break it in two inequalities each of the form $s_1 \vee \cdots \vee s_n \leq t_1 \vee \cdots \vee t_m$, and unltimately into a conjunction of equations of the form $s \leq t_1 \vee \cdots \vee t_m$, where s, t_j: monoid terms.

We proceed by example: $x^2 y \leq xy \vee yx$

$(x_1 \vee x_2)^2 y \leq (x_1 \vee x_2)y \vee y(x_1 \vee x_2)$

Equations

In an equation over $\{\cdot, 1, \vee\}$ we can distribute products over joins, then break it in two inequalities each of the form $s_1 \vee \cdots \vee s_n \leq t_1 \vee \cdots \vee t_m$, and unltimately into a conjunction of equations of the form $s \leq t_1 \vee \cdots \vee t_m$, where s, t_j: monoid terms.

We proceed by example: $x^2 y \leq xy \vee yx$

$(x_1 \vee x_2)^2 y \leq (x_1 \vee x_2)y \vee y(x_1 \vee x_2)$

$x_1^2 y \vee x_1 x_2 y \vee x_2 x_1 y \vee x_2^2 y \leq x_1 y \vee x_2 y \vee y x_1 \vee y x_2$

In an equation over $\{\cdot, 1, \vee\}$ we can distribute products over joins, then break it in two inequalities each of the form $s_1 \vee \cdots \vee s_n \leq t_1 \vee \cdots \vee t_m$, and unltimately into a conjunction of equations of the form $s \leq t_1 \vee \cdots \vee t_m$, where s, t_j: monoid terms.

We proceed by example: $x^2 y \leq xy \vee yx$

$(x_1 \vee x_2)^2 y \leq (x_1 \vee x_2) y \vee y(x_1 \vee x_2)$

$x_1^2 y \vee x_1 x_2 y \vee x_2 x_1 y \vee x_2^2 y \leq x_1 y \vee x_2 y \vee y x_1 \vee y x_2$

$x_1 x_2 y \leq x_1 y \vee x_2 y \vee y x_1 \vee y x_2$

Equations

In an equation over $\{\cdot, 1, \vee\}$ we can distribute products over joins, then break it in two inequalities each of the form $s_1 \vee \cdots \vee s_n \leq t_1 \vee \cdots \vee t_m$, and unltimately into a conjunction of equations of the form $s \leq t_1 \vee \cdots \vee t_m$, where s, t_j: monoid terms.

We proceed by example: $x^2 y \leq xy \vee yx$

$(x_1 \vee x_2)^2 y \leq (x_1 \vee x_2) y \vee y(x_1 \vee x_2)$

$x_1^2 y \vee x_1 x_2 y \vee x_2 x_1 y \vee x_2^2 y \leq x_1 y \vee x_2 y \vee y x_1 \vee y x_2$

$x_1 x_2 y \leq x_1 y \vee x_2 y \vee y x_1 \vee y x_2$

Therefore we consider inequalities of the form

$$(d) \quad x_1 \cdots x_n \leq \bigvee_{j=1}^{m} x_1^{d_j(1)} \cdots x_n^{d_j(n)},$$

where $d := \{d_1, ..., d_m\} \subseteq \mathbb{N}^n$.

Problems

Note that the inequality $x \leq x^2 \vee x^4$ also causes problems with the encoding: *it can lead to termination when it is not intended*.

Problems

Note that the inequality $x \leq x^2 \vee x^4$ also causes problems with the encoding: *it can lead to termination when it is not intended*.

For example if $\leq$ is the computation relation of M_{even} and $\leq'$ is the one where we also add the ambient inequality $x \leq x^2 \vee x^4$.

Problems

Note that the inequality $x \leq x^2 \vee x^4$ also causes problems with the encoding: *it can lead to termination when it is not intended*.

For example if $\leq$ is the computation relation of M_{even} and $\leq'$ is the one where we also add the ambient inequality $x \leq x^2 \vee x^4$.

On one hand we have $q_0 r^3 \not\leq q_F$ since 3 is odd.

Problems

Note that the inequality $x \leq x^2 \vee x^4$ also causes problems with the encoding: *it can lead to termination when it is not intended*.

For example if $\leq$ is the computation relation of M_{even} and $\leq'$ is the one where we also add the ambient inequality $x \leq x^2 \vee x^4$.

On one hand we have $q_0 r^3 \not\leq q_F$ since 3 is odd. On the other hand, $q_0 r^3 \leq' q_F$, because

Problems

Note that the inequality $x \leq x^2 \vee x^4$ also causes problems with the encoding: *it can lead to termination when it is not intended*.

For example if $\leq$ is the computation relation of M_{even} and $\leq'$ is the one where we also add the ambient inequality $x \leq x^2 \vee x^4$.

On one hand we have $q_0 r^3 \not\leq q_F$ since 3 is odd. On the other hand, $q_0 r^3 \leq' q_F$, because

$$q_0 r^3 = q_0 r^2 r \leq' q_0 r^2 r^2 \vee q_0 r^2 r^4 = q_0 r^4 \vee q_0 r^6 \leq' q_F,$$

since $q_0 r^4 \leq q_F$ and $q_0 r^6 \leq q_F$.

Problems

Note that the inequality $x \leq x^2 \vee x^4$ also causes problems with the encoding: *it can lead to termination when it is not intended*.

For example if $\leq$ is the computation relation of M_{even} and $\leq'$ is the one where we also add the ambient inequality $x \leq x^2 \vee x^4$.

On one hand we have $q_0 r^3 \not\leq q_F$ since 3 is odd. On the other hand, $q_0 r^3 \leq' q_F$, because

$$q_0 r^3 = q_0 r^2 r \leq' q_0 r^2 r^2 \vee q_0 r^2 r^4 = q_0 r^4 \vee q_0 r^6 \leq' q_F,$$

since $q_0 r^4 \leq q_F$ and $q_0 r^6 \leq q_F$.

However, this time we will show that unlike with $x \leq x^2$ we can get undecidability, by modifying the encoding a bit.

Idea

We assume that register r contains $n = s + t$ tokens and that $(d) : x \leq x^2 \vee x^4$ is applied to $t \neq 0$ of these tokens:

$$qr^n = qr^s r^t \leq' qr^s(r^{2t} \vee r^{4t}) = qr^{s+2t} \vee qr^{s+4t}$$

Idea

We assume that register r contains $n = s + t$ tokens and that $(d) : x \leq x^2 \vee x^4$ is applied to $t \neq 0$ of these tokens:

$$qr^n = qr^s r^t \leq' qr^s(r^{2t} \vee r^{4t}) = qr^{s+2t} \vee qr^{s+4t}$$

However, not both $s + 2t$ and $s + 4t$ can be powers of K, if $K \geq 3$.

Idea

We assume that register r contains $n = s + t$ tokens and that $(d) : x \leq x^2 \vee x^4$ is applied to $t \neq 0$ of these tokens:

$$qr^n = qr^s r^t \leq' qr^s(r^{2t} \vee r^{4t}) = qr^{s+2t} \vee qr^{s+4t}$$

However, not both $s + 2t$ and $s + 4t$ can be powers of K, if $K \geq 3$.

Idea: Make our intention that the contents of all registers are powers of K and that present instruction (blocks) always stay with powers of K.

Idea

We assume that register r contains $n = s + t$ tokens and that $(d) : x \leq x^2 \vee x^4$ is applied to $t \neq 0$ of these tokens:

$$qr^n = qr^s r^t \leq' qr^s(r^{2t} \vee r^{4t}) = qr^{s+2t} \vee qr^{s+4t}$$

However, not both $s + 2t$ and $s + 4t$ can be powers of K, if $K \geq 3$.

Idea: Make our intention that the contents of all registers are powers of K and that present instruction (blocks) always stay with powers of K. Then no application of the inequality (d) can help with termination.

We assume that register r contains $n = s + t$ tokens and that $(d) : x \leq x^2 \vee x^4$ is applied to $t \neq 0$ of these tokens:

$$qr^n = qr^s r^t \leq' qr^s(r^{2t} \vee r^{4t}) = qr^{s+2t} \vee qr^{s+4t}$$

However, not both $s + 2t$ and $s + 4t$ can be powers of K, if $K \geq 3$.

Idea: Make our intention that the contents of all registers are powers of K and that present instruction (blocks) always stay with powers of K. Then no application of the inequality (d) can help with termination.

To achieve that we implement the undecidable 2-counter machine M inside a machine M_K where contents of a register of the form n in M are replaced with K^n in M_K.

Idea

We assume that register r contains $n = s + t$ tokens and that $(d) : x \leq x^2 \vee x^4$ is applied to $t \neq 0$ of these tokens:

$$qr^n = qr^s r^t \leq' qr^s(r^{2t} \vee r^{4t}) = qr^{s+2t} \vee qr^{s+4t}$$

However, not both $s + 2t$ and $s + 4t$ can be powers of K, if $K \geq 3$.

Idea: Make our intention that the contents of all registers are powers of K and that present instruction (blocks) always stay with powers of K. Then no application of the inequality (d) can help with termination.

To achieve that we implement the undecidable 2-counter machine M inside a machine M_K where contents of a register of the form n in M are replaced with K^n in M_K. The instructions of M are replaced by programs in M_K.

Conditions for existence of K

If (d) implies a knotted inequality, then CRL $+ (d)$ is decidable.

Conditions for existence of K

If (d) implies a knotted inequality, then CRL $+ (d)$ is decidable.
E.g., if (d) is $xy \leq xy^2 \vee x^2y$, then

$$\mathsf{CRL} + (d) \models x^2 \leq x^3.$$

Conditions for existence of K

If (d) implies a knotted inequality, then CRL $+ (d)$ is decidable.
E.g., if (d) is $xy \leq xy^2 \vee x^2y$, then

$$\mathsf{CRL} + (d) \models x^2 \leq x^3.$$

A simple equation is called *prespinal* if there is a substitution that yields an inequality of the form

$$x_1 \cdots x_n \leq 1 \vee x_1^{k_{11}} \vee x_1^{k_{12}} x_2^{k_{22}} \vee \cdots \vee x_1^{k_{1n}} \cdots x_n^{k_{nn}}$$

where $k_{ii} \neq 0$, and where the term $1\vee$ could be missing.

If (d) implies a knotted inequality, then CRL $+ (d)$ is decidable.
E.g., if (d) is $xy \leq xy^2 \vee x^2y$, then

$$\mathsf{CRL} + (d) \models x^2 \leq x^3.$$

A simple equation is called *prespinal* if there is a substitution that yields an inequality of the form

$$x_1 \cdots x_n \leq 1 \vee x_1^{k_{11}} \vee x_1^{k_{12}} x_2^{k_{22}} \vee \cdots \vee x_1^{k_{1n}} \cdots x_n^{k_{nn}}$$

where $k_{ii} \neq 0$, and where the term $1\vee$ could be missing.

Theorem If a simple equation is not prespinal, then the word problem for its variety is undecidable.
Examples: $x \leq x^2 \vee x^3$,
$xy \leq x \vee x^2y \vee y^2$,
$xyzw \leq x^2yzw \vee x^3y^2z^2w^2$.

The machine M_K

Let M be the 2-register undecidable machine; we describe the new machine M_K. It has one more counter r_3. It contains all the states of M, it has three additional states z_1, z_2, z_3 in order to implement zero-tests and also contains states that are internal to its subprograms below. All copy instructions remain as they are, but we replace the add-one and substract-one instructions by multiply-by-K and divide-by-K programs.

$$\begin{array}{lcl} q \leq q'r & \Longrightarrow & qr^{\forall} \sqsubseteq q'r^{K\cdot\forall} \\ qr \leq q' & \Longrightarrow & qr^{\forall} \sqsubseteq q'r^{K\backslash\forall} \end{array}$$

The machine M_K

Let M be the 2-register undecidable machine; we describe the new machine M_K. It has one more counter r_3. It contains all the states of M, it has three additional states z_1, z_2, z_3 in order to implement zero-tests and also contains states that are internal to its subprograms below. All copy instructions remain as they are, but we replace the add-one and substract-one instructions by multiply-by-K and divide-by-K programs.

$$\begin{array}{lcllc} q & \leq & q'r & \Longrightarrow & qr^{\forall} \sqsubseteq q'r^{K\cdot\forall} \\ qr & \leq & q' & \Longrightarrow & qr^{\forall} \sqsubseteq q'r^{K\backslash\forall} \end{array}$$

We obtain, for each $q \in Q$,

$$qr_1^{n_1} r_2^{n_2} \leq_M q_f \iff qr_1^{K^{n_1}} r_2^{K^{n_2}} \leq_{M_K} q_F.$$

Programs

The **add-K program**: $q \sqsubseteq q' r_i^K$, for $i \in \{1, 2\}$, adds K tokens to register r_i, with input state q and output state q'.

$$q \leq a_1 r_i, \quad a_1 \leq a_2 r_j, \quad \cdots, \quad a_{K-1} \leq q' r_i.$$

The **add-K program**: $q \sqsubseteq q' r_i^K$, for $i \in \{1,2\}$, adds K tokens to register r_i, with input state q and output state q'.

$$q \leq a_1 r_i, \quad a_1 \leq a_2 r_j, \quad \cdots, \quad a_{K-1} \leq q' r_i.$$

The **transfer program**: $t_0 r_i^{\forall} \sqsubseteq q' r_j^{\forall}$, transfers all contents of register r_i to register r_j, with input state q and output state q'.

$$t_0 r_i \leq t_1, \qquad t_1 \leq t_0 r_j, \qquad t_0 r_i^{\emptyset} \sqsubseteq q'.$$

The **add-K program**: $q \sqsubseteq q' r_i^K$, for $i \in \{1,2\}$, adds K tokens to register r_i, with input state q and output state q'.

$$q \leq a_1 r_i, \quad a_1 \leq a_2 r_j, \quad \cdots, \quad a_{K-1} \leq q' r_i.$$

The **transfer program**: $t_0 r_i^{\forall} \sqsubseteq q' r_j^{\forall}$, transfers all contents of register r_i to register r_j, with input state q and output state q'.

$$t_0 r_i \leq t_1, \qquad t_1 \leq t_0 r_j, \qquad t_0 r_i^{\emptyset} \sqsubseteq q'.$$

The **multiply by K** program: $q r_i^{\forall} \sqsubseteq q' r_i^{K\cdot\forall}$, for each $i \in \{1,2\}$, multiplies the contents of r_i by K, with input state q and output state q'.

$$q r_3^{\emptyset} \sqsubseteq c, \quad c r_i \leq a_0, \quad a_0 \sqsubseteq c r_3^K, \quad c r_i^{\emptyset} \sqsubseteq t_0, \quad t_0 r_3^{\forall} \sqsubseteq q' r_i^{\forall}.$$

Programs

The **add-K program**: $q \sqsubseteq q' r_i^K$, for $i \in \{1,2\}$, adds K tokens to register r_i, with input state q and output state q'.

$$q \leq a_1 r_i, \quad a_1 \leq a_2 r_j, \quad \cdots, \quad a_{K-1} \leq q' r_i.$$

The **transfer program**: $t_0 r_i^{\forall} \sqsubseteq q' r_j^{\forall}$, transfers all contents of register r_i to register r_j, with input state q and output state q'.

$$t_0 r_i \leq t_1, \qquad t_1 \leq t_0 r_j, \qquad t_0 r_i^{\emptyset} \sqsubseteq q'.$$

The **multiply by K** program: $q r_i^{\forall} \sqsubseteq q' r_i^{K\cdot\forall}$, for each $i \in \{1,2\}$, multiplies the contents of r_i by K, with input state q and output state q'.

$$q r_3^{\emptyset} \sqsubseteq c, \quad c r_i \leq a_0, \quad a_0 \sqsubseteq c r_3^K, \quad c r_i^{\emptyset} \sqsubseteq t_0, \quad t_0 r_3^{\forall} \sqsubseteq q' r_i^{\forall}.$$

The **end program**: $q_f r_1 r_2 \sqsubseteq q_F$ transitions from the final state q_f of M to the final state q_F of M_K.

$$q_f r_1 \leq c_F, \quad c_F r_2 \leq q_F.$$

Proof idea

The inequality/glitch can modify the rules in a linear way.

Proof idea

The inequality/glitch can modify the rules in a linear way.

However, the contents in the register are stored in an exponential way.

Proof idea

The inequality/glitch can modify the rules in a linear way.

However, the contents in the register are stored in an exponential way.

We can prove that after an application of a non-prespinal inequality if the resulting registers in all configurations/joinands have powers of K, then the original contents of the registers were one of these configurations.

Proof idea

The inequality/glitch can modify the rules in a linear way.

However, the contents in the register are stored in an exponential way.

We can prove that after an application of a non-prespinal inequality if the resulting registers in all configurations/joinands have powers of K, then the original contents of the registers were one of these configurations.

This proves that applications of the inequality are actually redundant.

Proof idea

The inequality/glitch can modify the rules in a linear way.

However, the contents in the register are stored in an exponential way.

We can prove that after an application of a non-prespinal inequality if the resulting registers in all configurations/joinands have powers of K, then the original contents of the registers were one of these configurations.

This proves that applications of the inequality are actually redundant. The inequality is *admissible*, and the machine is *resilient* to applications of the inequality.

Proof idea

The inequality/glitch can modify the rules in a linear way.

However, the contents in the register are stored in an exponential way.

We can prove that after an application of a non-prespinal inequality if the resulting registers in all configurations/joinands have powers of K, then the original contents of the registers were one of these configurations.

This proves that applications of the inequality are actually redundant. The inequality is *admissible*, and the machine is *resilient* to applications of the inequality.

The proof that this works for non-prespinal inequalities proceeds by positive linear algebra.

Proof idea

Moving from multiplicative notation to additive one, the values of the exponents in the join in the RHS can be organized in a matrix R.

Proof idea

Moving from multiplicative notation to additive one, the values of the exponents in the join in the RHS can be organized in a matrix R.

The substitution that exhibits possible prespinality can be also written by a matrix S.

Proof idea

Moving from multiplicative notation to additive one, the values of the exponents in the join in the RHS can be organized in a matrix R.

The substitution that exhibits possible prespinality can be also written by a matrix S.

The spinal form corresponds to an upper triangular matrix T.

Proof idea

Moving from multiplicative notation to additive one, the values of the exponents in the join in the RHS can be organized in a matrix R.

The substitution that exhibits possible prespinality can be also written by a matrix S.

The spinal form corresponds to an upper triangular matrix T.

The problem reduces to whether there exist S and upper trianguālar T, both with natrual number coefficients, such that $SR = T$.

Proof idea

Moving from multiplicative notation to additive one, the values of the exponents in the join in the RHS can be organized in a matrix R.

The substitution that exhibits possible prespinality can be also written by a matrix S.

The spinal form corresponds to an upper triangular matrix T.

The problem reduces to whether there exist S and upper triangualar T, both with natrual number coefficients, such that $SR = T$.

By rearranging rows (relabeling variables) and columns (simultaneously in S and R) we can assume that both S and R are block-upper-triangular.

Proof idea

Moving from multiplicative notation to additive one, the values of the exponents in the join in the RHS can be organized in a matrix R.

The substitution that exhibits possible prespinality can be also written by a matrix S.

The spinal form corresponds to an upper triangular matrix T.

The problem reduces to whether there exist S and upper trianguallar T, both with natrual number coefficients, such that $SR = T$.

By rearranging rows (relabeling variables) and columns (simultaneously in S and R) we can assume that both S and R are block-upper-triangular.

Prespinality of R is proved to be equivalent to the existence of some positive solutions to systems of inequalities given by lower-right submatrices of R.

Proof idea

Moving from multiplicative notation to additive one, the values of the exponents in the join in the RHS can be organized in a matrix R.

The substitution that exhibits possible prespinality can be also written by a matrix S.

The spinal form corresponds to an upper triangular matrix T.

The problem reduces to whether there exist S and upper triangualar T, both with natrual number coefficients, such that $SR = T$.

By rearranging rows (relabeling variables) and columns (simultaneously in S and R) we can assume that both S and R are block-upper-triangular.

Prespinality of R is proved to be equivalent to the existence of some positive solutions to systems of inequalities given by lower-right submatrices of R.

We prove that there exists a large enough value of K exhibiting the admissibiity of non-prispinal rules.

Proof idea

Moving from multiplicative notation to additive one, the values of the exponents in the join in the RHS can be organized in a matrix R.

The substitution that exhibits possible prespinality can be also written by a matrix S.

The spinal form corresponds to an upper triangular matrix T.

The problem reduces to whether there exist S and upper triangualar T, both with natrual number coefficients, such that $SR = T$.

By rearranging rows (relabeling variables) and columns (simultaneously in S and R) we can assume that both S and R are block-upper-triangular.

Prespinality of R is proved to be equivalent to the existence of some positive solutions to systems of inequalities given by lower-right submatrices of R.

We prove that there exists a large enough value of K exhibiting the admissibiity of non-prispinal rules. This is done by moving from $\mathbb{N}$ to $\mathbb{R}$ and using the theorem of alternatives of positive linear algebra.

Undecidable equational theory

We can encode the instructions of a machine as a single term θ using the full signature of of CRL via

$$\theta := 1 \wedge \bigwedge_{(a \leq b) \in P} a \to b.$$

Undecidable equational theory

We can encode the instructions of a machine as a single term θ using the full signature of of CRL via

$$\theta := 1 \wedge \bigwedge_{(a \leq b) \in P} a \to b.$$

If an inequality (d) implies an equation of the form

$$(x \wedge 1)^k \leq \bigvee_{i=1}^{n} (x \wedge 1)^{k+c_i} \tag{1}$$

for some $n \geq 1$ and $k, c_1, ..., c_n \geq 1$, then we can reduce the decidability of the quasiequational equational theory (or at least of the part we have shown to be undecidable) to that of the equational theory.

Undecidable equational theory

We can encode the instructions of a machine as a single term θ using the full signature of of CRL via

$$\theta := 1 \wedge \bigwedge_{(a \leq b) \in P} a \to b.$$

If an inequality (d) implies an equation of the form

$$(x \wedge 1)^k \leq \bigvee_{i=1}^{n} (x \wedge 1)^{k+c_i} \tag{1}$$

for some $n \geq 1$ and $k, c_1, ..., c_n \geq 1$, then we can reduce the decidability of the quasiequational equational theory (or at least of the part we have shown to be undecidable) to that of the equational theory.

For example the equational theory of $x \leq x^2 \vee x^3$ is undecidable and the same holds for equations of the form $x^k \leq x^{k+c_1} \vee \cdots x^{k+c_n}$.