Symbolic Computation in Maude: Some Tapas

José Meseguer

University of Illinois at Urbana-Champaign

What is Maude?

Q: What is Maude?

What is Maude?

Q: What is Maude?

A: A high-performance declarative language whose modules are theories in rewriting logic.

What is Maude?

Q: What is Maude?

A: A high-performance declarative language whose modules are theories in rewriting logic.

Q: What is rewriting logic?

What is Maude?

Q: What is Maude?

A: A high-performance declarative language whose modules are theories in rewriting logic.

Q: What is rewriting logic?

A: A simple, yet expressive, computational logic to specify and program concurrent systems as rewrite theories.

What is Maude?

Q: What is Maude?

A: A high-performance declarative language whose modules are theories in rewriting logic.

Q: What is rewriting logic?

A: A simple, yet expressive, computational logic to specify and program concurrent systems as rewrite theories.

Q: What is a rewrite theory?

What is Maude?

Q: What is Maude?

A: A high-performance declarative language whose modules are theories in rewriting logic.

Q: What is rewriting logic?

A: A simple, yet expressive, computational logic to specify and program concurrent systems as rewrite theories.

Q: What is a rewrite theory?

A: A triple $\mathcal{R} = (\Sigma, \mathit{EUB}, R)$ where:

What is Maude?

Q: What is Maude?

A: A high-performance declarative language whose modules are theories in rewriting logic.

Q: What is rewriting logic?

A: A simple, yet expressive, computational logic to specify and program concurrent systems as rewrite theories.

Q: What is a rewrite theory?

A: A triple $\mathcal{R} = (\Sigma, \mathit{EUB}, R)$ where:

- (Σ, EUB) is an equational theory specifying the concurrent system's states as an algebraic data type.

What is Maude?

Q: What is Maude?

A: A high-performance declarative language whose modules are theories in rewriting logic.

Q: What is rewriting logic?

A: A simple, yet expressive, computational logic to specify and program concurrent systems as rewrite theories.

Q: What is a rewrite theory?

A: A triple $\mathcal{R} = (\Sigma, EUB, R)$ where:

- (Σ, EUB) is an equational theory specifying the concurrent system's states as an algebraic data type.
- R are rewrite rules specifying the system's atomic transitions.

What is Maude?

Q: What is Maude?

A: A high-performance declarative language whose modules are theories in rewriting logic.

Q: What is rewriting logic?

A: A simple, yet expressive, computational logic to specify and program concurrent systems as rewrite theories.

Q: What is a rewrite theory?

A: A triple $\mathcal{R} = (\Sigma, EUB, R)$ where:

- (Σ, EUB) is an equational theory specifying the concurrent system's states as an algebraic data type.
- R are rewrite rules specifying the system's atomic transitions.
- $\mathit{Concurrent\ \ Computation} = \mathit{Deduction}$ in $\mathcal{R}$

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, \mathit{EUB}, R)$ becomes an equational theory,

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, EUB, R)$ becomes an equational theory, Maude has a functional sublanguage whose modules:

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, EUB, R)$ becomes an equational theory, Maude has a functional sublanguage whose modules:

$$\textbf{fmod } (\Sigma, EUB) \textbf{ endfm}$$

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, EUB, R)$ becomes an equational theory, Maude has a functional sublanguage whose modules:

$$\textbf{fmod}\ (\Sigma, EUB)\ \textbf{endfm}$$

called functional modules are such that:

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, EUB, R)$ becomes an equational theory, Maude has a functional sublanguage whose modules:

$$\textbf{fmod}\ (\Sigma, EUB)\ \textbf{endfm}$$

called functional modules are such that:

- $B \subseteq \{A, C, U\}$ is any combination of associativity (A) and/or commutativity (C) and/or identity (U) axioms.

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, EUB, R)$ becomes an equational theory, Maude has a functional sublanguage whose modules:

$$\textbf{fmod } (\Sigma, EUB) \textbf{ endfm}$$

called functional modules are such that:

- $B \subseteq \{A, C, U\}$ is any combination of associativity (A) and/or commutativity (C) and/or identity (U) axioms.
- The equations E are convergent modulo the axioms B.

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, EUB, R)$ becomes an equational theory, Maude has a functional sublanguage whose modules:

$$\textbf{fmod } (\Sigma, EUB) \textbf{ endfm}$$

called functional modules are such that:

- $B \subseteq \{A, C, U\}$ is any combination of associativity (A) and/or commutativity (C) and/or identity (U) axioms.
- The equations E are convergent modulo the axioms B.

When $R \neq \emptyset$ we have general system modules of the form:

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, EUB, R)$ becomes an equational theory, Maude has a functional sublanguage whose modules:

$$\textbf{fmod } (\Sigma, EUB) \textbf{ endfm}$$

called functional modules are such that:

- $B \subseteq \{A, C, U\}$ is any combination of associativity (A) and/or commutativity (C) and/or identity (U) axioms.
- The equations E are convergent modulo the axioms B.

When $R \neq \emptyset$ we have general system modules of the form:

$$\textbf{mod } (\Sigma, EUB, R) \textbf{ endm}$$

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, EUB, R)$ becomes an equational theory, Maude has a functional sublanguage whose modules:

$$\textbf{fmod } (\Sigma, EUB) \textbf{ endfm}$$

called functional modules are such that:

- $B \subseteq \{A, C, U\}$ is any combination of associativity (A) and/or commutativity (C) and/or identity (U) axioms.
- The equations E are convergent modulo the axioms B.

When $R \neq \emptyset$ we have general system modules of the form:

$$\textbf{mod } (\Sigma, EUB, R) \textbf{ endm}$$

specifying concurrent systems, where:

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, E \cup B, R)$ becomes an equational theory, Maude has a functional sublanguage whose modules:

$$\textbf{fmod } (\Sigma, E \cup B) \textbf{ endfm}$$

called functional modules are such that:

- $B \subseteq \{A, C, U\}$ is any combination of associativity (A) and/or commutativity (C) and/or identity (U) axioms.
- The equations E are convergent modulo the axioms B.

When $R \neq \emptyset$ we have general system modules of the form:

$$\textbf{mod } (\Sigma, E \cup B, R) \textbf{ endm}$$

specifying concurrent systems, where:

- The equations E are convergent modulo the axioms B.

What is Maude? (II)

Since when $R = \emptyset$, $\mathcal{R} = (\Sigma, EUB, R)$ becomes an equational theory, Maude has a functional sublanguage whose modules:

$$\textbf{fmod } (\Sigma, EUB) \textbf{ endfm}$$

called functional modules are such that:

- $B \subseteq \{A, C, U\}$ is any combination of associativity (A) and/or commutativity (C) and/or identity (U) axioms.
- The equations E are convergent modulo the axioms B.

When $R \neq \emptyset$ we have general system modules of the form:

$$\textbf{mod } (\Sigma, EUB, R) \textbf{ endm}$$

specifying concurrent systems, where:

- The equations E are convergent modulo the axioms B.
- The rules R are coherent with E modulo B.

Standard computation is performed by rewriting with equations E and rules R modulo B.

Standard computation is performed by rewriting with equations E and rules R modulo B.

But

Symbolic Computation in Maude

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables,

Symbolic Computation in Maude

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables, including:

Symbolic Computation in Maude

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables, including:

1. B-**Unification** (for any $B \subseteq \{A, C, U\}$)

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables, including:

1. B-**Unification** (for any $B \subseteq \{A, C, U\}$)
2. B-**Generalization**

Symbolic Computation in Maude

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables, including:

1. B-**Unification** (for any $B \subseteq \{A, C, U\}$)
2. B-**Generalization**
3. E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$,

Symbolic Computation in Maude

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables, including:

1. B-**Unification** (for any $B \subseteq \{A, C, U\}$)
2. B-**Generalization**
3. E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$, which is finitary iff $(\Sigma, E \cup B)$ has the finite variant property (FVP)

Symbolic Computation in Maude

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables, including:

1. B-**Unification** (for any $B \subseteq \{A, C, U\}$)
2. B-**Generalization**
3. E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$, which is finitary iff $(\Sigma, E \cup B)$ has the finite variant property (FVP)
4. $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$,

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables, including:

1. B-**Unification** (for any $B \subseteq \{A, C, U\}$)
2. B-**Generalization**
3. E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$, which is finitary iff $(\Sigma, E \cup B)$ has the finite variant property (FVP)
4. $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$, which is finitary iff $(\Sigma, E \cup B)$ is FVP

Symbolic Computation in Maude

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables, including:

1. B-**Unification** (for any $B \subseteq \{A, C, U\}$)
2. B-**Generalization**
3. E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$, which is finitary iff $(\Sigma, E \cup B)$ has the finite variant property (FVP)
4. $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$, which is finitary iff $(\Sigma, E \cup B)$ is FVP
5. **Domain-Specific SMT-Solving**, thanks to CVC4 and Yices interfaces

Symbolic Computation in Maude

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables, including:

1. B-**Unification** (for any $B \subseteq \{A, C, U\}$)
2. B-**Generalization**
3. E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$, which is finitary iff $(\Sigma, E \cup B)$ has the finite variant property (FVP)
4. $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$, which is finitary iff $(\Sigma, E \cup B)$ is FVP
5. **Domain-Specific SMT-Solving**, thanks to CVC4 and Yices interfaces
6. **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$

Standard computation is performed by rewriting with equations E and rules R modulo B.

But Maude also supports a variety of symbolic reasoning taks with terms and formulas involving logical variables, including:

1. B-**Unification** (for any $B \subseteq \{A, C, U\}$)
2. B-**Generalization**
3. E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$, which is finitary iff $(\Sigma, E \cup B)$ has the finite variant property (FVP)
4. $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$, which is finitary iff $(\Sigma, E \cup B)$ is FVP
5. **Domain-Specific SMT-Solving**, thanks to CVC4 and Yices interfaces
6. **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$
7. **Symbolic Reachabilty Analysis** of any system module **mod** (Σ, EUB, R) **endm** with $(\Sigma, E \cup B)$ FVP.

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$
- **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$.

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$
- **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$.
- **Symbolic Reachabilty Analysis** of any system module **mod** (Σ, EUB, R) **endm** with $(\Sigma, E \cup B)$ FVP.

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$
- **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$.
- **Symbolic Reachabilty Analysis** of any system module **mod** (Σ, EUB, R) **endm** with $(\Sigma, E \cup B)$ FVP.

Why this focus?

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$
- **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$.
- **Symbolic Reachabilty Analysis** of any system module **mod** (Σ, EUB, R) **endm** with $(\Sigma, E \cup B)$ FVP.

Why this focus? Because these methods are both theory-generic (apply to an infinite class of theories), and

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$
- **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$.
- **Symbolic Reachabilty Analysis** of any system module **mod** (Σ, EUB, R) **endm** with $(\Sigma, E \cup B)$ FVP.

Why this focus? Because these methods are both theory-generic (apply to an infinite class of theories), and user-definable.

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$
- **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$.
- **Symbolic Reachabilty Analysis** of any system module **mod** (Σ, EUB, R) **endm** with $(\Sigma, E \cup B)$ FVP.

Why this focus? Because these methods are both theory-generic (apply to an infinite class of theories), and user-definable.

These methods make symbolic reasoning user-extensible way beyond the fixed bag of tricks of domain-specific methods

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$
- **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$.
- **Symbolic Reachabilty Analysis** of any system module **mod** (Σ, EUB, R) **endm** with $(\Sigma, E \cup B)$ FVP.

Why this focus? Because these methods are both theory-generic (apply to an infinite class of theories), and user-definable.

These methods make symbolic reasoning user-extensible way beyond the fixed bag of tricks of domain-specific methods such as B-unification or domain-specific SMT solving.

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$
- **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$.
- **Symbolic Reachabilty Analysis** of any system module **mod** (Σ, EUB, R) **endm** with $(\Sigma, E \cup B)$ FVP.

Why this focus? Because these methods are both theory-generic (apply to an infinite class of theories), and user-definable.

These methods make symbolic reasoning user-extensible way beyond the fixed bag of tricks of domain-specific methods such as B-unification or domain-specific SMT solving.

To wet you appetite,

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$
- **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$.
- **Symbolic Reachabilty Analysis** of any system module **mod** (Σ, EUB, R) **endm** with $(\Sigma, E \cup B)$ FVP.

Why this focus? Because these methods are both theory-generic (apply to an infinite class of theories), and user-definable.

These methods make symbolic reasoning user-extensible way beyond the fixed bag of tricks of domain-specific methods such as B-unification or domain-specific SMT solving.

To wet you appetite, I will illustrate these methods with examples:

Symbolic Computation in Maude (II)

In this talk I will focus on four Maude-supported methods:

- E, B-**Variants** of a term t in a convergent $(\Sigma, E \cup B)$
- $E \cup B$-**Unification** for any convergent $(\Sigma, E \cup B)$
- **Theory-Generic SMT-Solving** for any OS-compact FVP theory $(\Sigma, E \cup B)$.
- **Symbolic Reachabilty Analysis** of any system module **mod** (Σ, EUB, R) **endm** with $(\Sigma, E \cup B)$ FVP.

Why this focus? Because these methods are both theory-generic (apply to an infinite class of theories), and user-definable.

These methods make symbolic reasoning user-extensible way beyond the fixed bag of tricks of domain-specific methods such as B-unification or domain-specific SMT solving.

To wet you appetite, I will illustrate these methods with examples: some tapas.

Variants in a Nutshell

Consider an equational theory $(\Sigma, E \cup B)$,

Variants in a Nutshell

Consider an equational theory $(\Sigma, E \cup B)$, with E convergent applied modulo B.

Variants in a Nutshell

Consider an equational theory $(\Sigma, E \cup B)$, with E convergent applied modulo B.

Can think of a Σ-term t with variables as a functional expression

Variants in a Nutshell

Consider an equational theory $(\Sigma, E \cup B)$, with E convergent applied modulo B.

Can think of a Σ-term t with variables as a functional expression to be symbolically evaluated with E modulo B.

Variants in a Nutshell

Consider an equational theory $(\Sigma, E \cup B)$, with E convergent applied modulo B.

Can think of a Σ-term t with variables as a functional expression to be symbolically evaluated with E modulo B.

The Comon-Delaune notion of the E, B-variants of t

Variants in a Nutshell

Consider an equational theory $(\Sigma, E \cup B)$, with E convergent applied modulo B.

Can think of a Σ-term t with variables as a functional expression to be symbolically evaluated with E modulo B.

The Comon-Delaune notion of the E, B-variants of t describes the different normalized symbolic results to which t can be symbolically evaluated.

Variants in a Nutshell

Consider an equational theory $(\Sigma, E \cup B)$, with E convergent applied modulo B.

Can think of a Σ-term t with variables as a functional expression to be symbolically evaluated with E modulo B.

The Comon-Delaune notion of the E, B-variants of t describes the different normalized symbolic results to which t can be symbolically evaluated.

Symbolic evaluation is performed by

Variants in a Nutshell

Consider an equational theory $(\Sigma, E \cup B)$, with E convergent applied modulo B.

Can think of a Σ-term t with variables as a functional expression to be symbolically evaluated with E modulo B.

The Comon-Delaune notion of the E, B-variants of t describes the different normalized symbolic results to which t can be symbolically evaluated.

Symbolic evaluation is performed by narrowing t with rules E

Variants in a Nutshell

Consider an equational theory $(\Sigma, E \cup B)$, with E convergent applied modulo B.

Can think of a Σ-term t with variables as a functional expression to be symbolically evaluated with E modulo B.

The Comon-Delaune notion of the E, B-variants of t describes the different normalized symbolic results to which t can be symbolically evaluated.

Symbolic evaluation is performed by narrowing t with rules E modulo axioms B.

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \leadsto^{\sigma}_{E,B} t'$

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \rightsquigarrow^{\sigma}_{E,B} t'$ is defined iff there is:

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \leadsto^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \leadsto^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in E; and

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \rightsquigarrow^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in E; and
- a B-unifier σ

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \leadsto^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in E; and
- a B-unifier σ such that $\sigma(t|_p) =_B \sigma(l)$, and $t' = \sigma(t[r]_p)$.

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \leadsto^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in E; and
- a B-unifier σ such that $\sigma(t|_p) =_B \sigma(l)$, and $t' = \sigma(t[r]_p)$.

A complete set of variants of t

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \rightsquigarrow^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in E; and
- a B-unifier σ such that $\sigma(t|_p) =_B \sigma(l)$, and $t' = \sigma(t[r]_p)$.

A complete set of variants of t can be computed as those t' such that $t \overset{\theta}{\rightsquigarrow}^{*}_{E,B} t'$ and t' is in E, B-normal form.

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \leadsto^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in E; and
- a B-unifier σ such that $\sigma(t|_p) =_B \sigma(l)$, and $t' = \sigma(t[r]_p)$.

A complete set of variants of t can be computed as those t' such that $t \overset{\theta}{\leadsto}^{*}_{E,B} t'$ and t' is in E, B-normal form.

Folding variant narrowing

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \leadsto^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in E; and
- a B-unifier σ such that $\sigma(t|_p) =_B \sigma(l)$, and $t' = \sigma(t[r]_p)$.

A complete set of variants of t can be computed as those t' such that $t \overset{\theta}{\leadsto}{}^{*}_{E,B} t'$ and t' is in E, B-normal form.

Folding variant narrowing is a strategy to compute a complete set of most general variants using variant subsumption.

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \leadsto^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in E; and
- a B-unifier σ such that $\sigma(t|_p) =_B \sigma(l)$, and $t' = \sigma(t[r]_p)$.

A complete set of variants of t can be computed as those t' such that $t \overset{\theta}{\leadsto}{}^{*}_{E,B} t'$ and t' is in E, B-normal form.

Folding variant narrowing is a strategy to compute a complete set of most general variants using variant subsumption.

$(\Sigma, E \cup B)$ has the finite variant property (FVP)

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \leadsto^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in E; and
- a B-unifier σ such that $\sigma(t|_p) =_B \sigma(l)$, and $t' = \sigma(t[r]_p)$.

A complete set of variants of t can be computed as those t' such that $t \overset{\theta}{\leadsto}{}^{*}_{E,B} t'$ and t' is in E, B-normal form.

Folding variant narrowing is a strategy to compute a complete set of most general variants using variant subsumption.

$(\Sigma, E \cup B)$ has the finite variant property (FVP) iff folding variant narrowing terminates for any term t with a finite set of most general variants.

Equational Narrowing in a Nutshell

For $(\Sigma, E \cup B)$ as above, the narrowing relation $t \rightsquigarrow^{\sigma}_{E,B} t'$ is defined iff there is:

- a non-variable position $p \in Pos(t)$;
- a rule $l \to r$ in E; and
- a B-unifier σ such that $\sigma(t|_p) =_B \sigma(l)$, and $t' = \sigma(t[r]_p)$.

A complete set of variants of t can be computed as those t' such that $t \overset{\theta}{\rightsquigarrow}^{*}_{E,B} t'$ and t' is in E, B-normal form.

Folding variant narrowing is a strategy to compute a complete set of most general variants using variant subsumption.

$(\Sigma, E \cup B)$ has the finite variant property (FVP) iff folding variant narrowing terminates for any term t with a finite set of most general variants. FVP is semi-decidable and easily checkable in Maude when it holds.

A non-FVP Example: Peano Naturals

Consider the problem of narrowing (i.e., symbolically executing) the term $x + y$ with the equations E defining addition of naturals in Peano notation (and no axioms: $B = \emptyset$)

A non-FVP Example: Peano Naturals

Consider the problem of narrowing (i.e., symbolically executing) the term $x + y$ with the equations E defining addition of naturals in Peano notation (and no axioms: $B = \emptyset$)

- $0 + y = y$
- $s(x) + y = s(x + y)$

A non-FVP Example: Peano Naturals

Consider the problem of narrowing (i.e., symbolically executing) the term $x + y$ with the equations E defining addition of naturals in Peano notation (and no axioms: $B = \emptyset$)

- $0 + y = y$
- $s(x) + y = s(x + y)$

This example is not FVP.

A non-FVP Example: Peano Naturals

Consider the problem of narrowing (i.e., symbolically executing) the term $x + y$ with the equations E defining addition of naturals in Peano notation (and no axioms: $B = \emptyset$)

- $0 + y = y$
- $s(x) + y = s(x + y)$

This example is not FVP. For example, the term $x + y$ has an infinite set of incomparable variants, including:

A non-FVP Example: Peano Naturals

Consider the problem of narrowing (i.e., symbolically executing) the term $x + y$ with the equations E defining addition of naturals in Peano notation (and no axioms: $B = \emptyset$)

- $0 + y = y$
- $s(x) + y = s(x + y)$

This example is not FVP. For example, the term $x + y$ has an infinite set of incomparable variants, including:

$$y, \quad s(x' + y), \quad s(y), \quad s(s(x'' + y)), \quad s(s(y), \ldots$$

A non-FVP Example: Peano Naturals

Consider the problem of narrowing (i.e., symbolically executing) the term $x + y$ with the equations E defining addition of naturals in Peano notation (and no axioms: $B = \emptyset$)

- $0 + y = y$
- $s(x) + y = s(x + y)$

This example is not FVP. For example, the term $x + y$ has an infinite set of incomparable variants, including:

$$y, \quad s(x' + y), \quad s(y), \quad s(s(x'' + y)), \quad s(s(y), \ldots$$

Maude can incrementally compute all the variants of $x + y$,

A non-FVP Example: Peano Naturals

Consider the problem of narrowing (i.e., symbolically executing) the term $x + y$ with the equations E defining addition of naturals in Peano notation (and no axioms: $B = \emptyset$)

- $0 + y = y$
- $s(x) + y = s(x + y)$

This example is not FVP. For example, the term $x + y$ has an infinite set of incomparable variants, including:

$$y, \quad s(x' + y), \quad s(y), \quad s(s(x'' + y)), \quad s(s(y), \ldots$$

Maude can incrementally compute all the variants of $x + y$, but it never terminates.

An FVP Example: Presburger Arithmetic

Let $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ be the Presburger arithmetic FVP two-sorted equational specification (*Nat* and *Bool*) with:

An FVP Example: Presburger Arithmetic

Let $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ be the Presburger arithmetic FVP two-sorted equational specification (*Nat* and *Bool*) with:

- $\Sigma = \{0, 1, +, >, \top, \bot\}$,

An FVP Example: Presburger Arithmetic

Let $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ be the Presburger arithmetic FVP two-sorted equational specification (*Nat* and *Bool*) with:

- $\Sigma = \{0, 1, +, >, \top, \bot\}$,
- E two equations, defining $>$, oriented as rewrite rules $m + n + 1 > n \to \top$ and $n > n + m \to \bot$, and

An FVP Example: Presburger Arithmetic

Let $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ be the Presburger arithmetic FVP two-sorted equational specification (*Nat* and *Bool*) with:

- $\Sigma = \{0, 1, +, >, \top, \bot\}$,
- E two equations, defining $>$, oriented as rewrite rules $m + n + 1 > n \to \top$ and $n > n + m \to \bot$, and
- ACU the axioms of associativity commutativity (AC) and unit 0 (U) for +.

An FVP Example: Presburger Arithmetic

Let $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ be the Presburger arithmetic FVP two-sorted equational specification (*Nat* and *Bool*) with:

- $\Sigma = \{0, 1, +, >, \top, \bot\}$,
- E two equations, defining $>$, oriented as rewrite rules $m + n + 1 > n \to \top$ and $n > n + m \to \bot$, and
- ACU the axioms of associativity commutativity (AC) and unit 0 (U) for +.

The initial algebra of $\mathcal{N}_{+,>}$ is the Presburger natural numbers.

An FVP Example: Presburger Arithmetic

Let $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ be the Presburger arithmetic FVP two-sorted equational specification (*Nat* and *Bool*) with:

- $\Sigma = \{0, 1, +, >, \top, \bot\}$,
- E two equations, defining $>$, oriented as rewrite rules $m + n + 1 > n \to \top$ and $n > n + m \to \bot$, and
- ACU the axioms of associativity commutativity (AC) and unit 0 (U) for $+$.

The initial algebra of $\mathcal{N}_{+,>}$ is the Presburger natural numbers.

Folding variant narrowing computes the following three most general variants of the term $x > y$:

An FVP Example: Presburger Arithmetic

Let $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ be the Presburger arithmetic FVP two-sorted equational specification (*Nat* and *Bool*) with:

- $\Sigma = \{0, 1, +, >, \top, \bot\}$,
- E two equations, defining $>$, oriented as rewrite rules $m + n + 1 > n \to \top$ and $n > n + m \to \bot$, and
- ACU the axioms of associativity commutativity (AC) and unit 0 (U) for +.

The initial algebra of $\mathcal{N}_{+,>}$ is the Presburger natural numbers.

Folding variant narrowing computes the following three most general variants of the term $x > y$:

- $x > y$ itself, with identity substitution

An FVP Example: Presburger Arithmetic

Let $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ be the Presburger arithmetic FVP two-sorted equational specification (*Nat* and *Bool*) with:

- $\Sigma = \{0, 1, +, >, \top, \bot\}$,
- E two equations, defining $>$, oriented as rewrite rules $m + n + 1 > n \to \top$ and $n > n + m \to \bot$, and
- ACU the axioms of associativity commutativity (AC) and unit 0 (U) for +.

The initial algebra of $\mathcal{N}_{+,>}$ is the Presburger natural numbers.

Folding variant narrowing computes the following three most general variants of the term $x > y$:

- $x > y$ itself, with identity substitution
- $\top$, with substitution $\{x \mapsto 1 + n + m, y \mapsto n\}$,

An FVP Example: Presburger Arithmetic

Let $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ be the Presburger arithmetic FVP two-sorted equational specification (*Nat* and *Bool*) with:

- $\Sigma = \{0, 1, +, >, \top, \bot\}$,
- E two equations, defining $>$, oriented as rewrite rules $m + n + 1 > n \to \top$ and $n > n + m \to \bot$, and
- ACU the axioms of associativity commutativity (AC) and unit 0 (U) for +.

The initial algebra of $\mathcal{N}_{+,>}$ is the Presburger natural numbers.

Folding variant narrowing computes the following three most general variants of the term $x > y$:

- $x > y$ itself, with identity substitution
- $\top$, with substitution $\{x \mapsto 1 + n + m, y \mapsto n\}$,
- $\bot$ with substitution $\{x \mapsto n, y \mapsto n + m\}$.

Variant Unification as Folding Variant Narrowing

Unification modulo Presburger Arithmetic $\mathcal{N}_{+,>}$ is computed by folding variant narrowing by just:

Variant Unification as Folding Variant Narrowing

Unification modulo Presburger Arithmetic $\mathcal{N}_{+,>}$ is computed by folding variant narrowing by just:

- adding a binary operator $\_ \equiv \_$ for solving equations and

Unification modulo Presburger Arithmetic $\mathcal{N}_{+,>}$ is computed by folding variant narrowing by just:

- adding a binary operator $\_ \equiv \_$ for solving equations and
- the single rewrite rule $x \equiv x \to \top$.

Variant Unification as Folding Variant Narrowing

Unification modulo Presburger Arithmetic $\mathcal{N}_{+,>}$ is computed by folding variant narrowing by just:

- adding a binary operator $\_ \equiv \_$ for solving equations and
- the single rewrite rule $x \equiv x \to \top$.

In general, the $E \cup B$-variant unifiers of two terms u, v modulo $\mathcal{N}_{+,>}$

Variant Unification as Folding Variant Narrowing

Unification modulo Presburger Arithmetic $\mathcal{N}_{+,>}$ is computed by folding variant narrowing by just:

- adding a binary operator $\_ \equiv \_$ for solving equations and
- the single rewrite rule $x \equiv x \to \top$.

In general, the $E \cup B$-variant unifiers of two terms u, v modulo $\mathcal{N}_{+,>}$ are precisely the substitutions θ associated to the variants of the form $\top$ of the term $u \equiv v$,

Variant Unification as Folding Variant Narrowing

Unification modulo Presburger Arithmetic $\mathcal{N}_{+,>}$ is computed by folding variant narrowing by just:

- adding a binary operator $\_ \equiv \_$ for solving equations and
- the single rewrite rule $x \equiv x \to \top$.

In general, the $E \cup B$-variant unifiers of two terms u, v modulo $\mathcal{N}_{+,>}$ are precisely the substitutions θ associated to the variants of the form $\top$ of the term $u \equiv v$, computed by folding variant narrowing sequences $u \equiv v \overset{\theta}{\rightsquigarrow}{}^{*}_{E,B} \top$.

Variant Unification as Folding Variant Narrowing

Unification modulo Presburger Arithmetic $\mathcal{N}_{+,>}$ is computed by folding variant narrowing by just:

- adding a binary operator $\_ \equiv \_$ for solving equations and
- the single rewrite rule $x \equiv x \to \top$.

In general, the $E \cup B$-variant unifiers of two terms u, v modulo $\mathcal{N}_{+,>}$ are precisely the substitutions θ associated to the variants of the form $\top$ of the term $u \equiv v$, computed by folding variant narrowing sequences $u \equiv v \overset{\theta}{\leadsto}{}^{*}_{E,B} \top$.

Since $\mathcal{N}_{+,>}$ is FVP, there is a finite number of variants of $u \equiv v$,

Variant Unification as Folding Variant Narrowing

Unification modulo Presburger Arithmetic $\mathcal{N}_{+,>}$ is computed by folding variant narrowing by just:

- adding a binary operator $\_ \equiv \_$ for solving equations and
- the single rewrite rule $x \equiv x \to \top$.

In general, the $E \cup B$-variant unifiers of two terms u, v modulo $\mathcal{N}_{+,>}$ are precisely the substitutions θ associated to the variants of the form $\top$ of the term $u \equiv v$, computed by folding variant narrowing sequences $u \equiv v \overset{\theta}{\rightsquigarrow}{}^{*}_{E,B} \top$.

Since $\mathcal{N}_{+,>}$ is FVP, there is a finite number of variants of $u \equiv v$, i.e., Presburger Arithmetic $\mathcal{N}_{+,>}$-unification is finitary.

Variant Unification as Folding Variant Narrowing

Unification modulo Presburger Arithmetic $\mathcal{N}_{+,>}$ is computed by folding variant narrowing by just:

- adding a binary operator $\_ \equiv \_$ for solving equations and
- the single rewrite rule $x \equiv x \to \top$.

In general, the $E \cup B$-variant unifiers of two terms u, v modulo $\mathcal{N}_{+,>}$ are precisely the substitutions θ associated to the variants of the form $\top$ of the term $u \equiv v$, computed by folding variant narrowing sequences $u \equiv v \overset{\theta}{\leadsto}^{*}_{E,B} \top$.

Since $\mathcal{N}_{+,>}$ is FVP, there is a finite number of variants of $u \equiv v$, i.e., Presburger Arithmetic $\mathcal{N}_{+,>}$-unification is finitary.

For example, $x > y \equiv y > x$ has the single unifier $\{x \mapsto y\}$ modulo $\mathcal{N}_{+,>}$.

Constructor Variants and Constructor Unifiers

In $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ the predicate $>$ is a defined symbol: it evaluates to either $\top$ or $\bot$.

Constructor Variants and Constructor Unifiers

In $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ the predicate $>$ is a defined symbol: it evaluates to either $\top$ or $\bot$. Instead, the other operators $\Omega = \{0, 1, +, \top, \bot\}$ are constructor symbols.

Constructor Variants and Constructor Unifiers

In $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ the predicate $>$ is a defined symbol: it evaluates to either $\top$ or $\bot$. Instead, the other operators $\Omega = \{0, 1, +, \top, \bot\}$ are constructor symbols.

A constructor variant

Constructor Variants and Constructor Unifiers

In $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ the predicate $>$ is a defined symbol: it evaluates to either $\top$ or $\bot$. Instead, the other operators $\Omega = \{0, 1, +, \top, \bot\}$ are constructor symbols.

A constructor variant is variant that is a constructor term.

Constructor Variants and Constructor Unifiers

In $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ the predicate $>$ is a defined symbol: it evaluates to either $\top$ or $\bot$. Instead, the other operators $\Omega = \{0, 1, +, \top, \bot\}$ are constructor symbols.

A constructor variant is variant that is a constructor term. For example, $\top$ and $\bot$ are constructor variants of $x > y$,

Constructor Variants and Constructor Unifiers

In $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ the predicate $>$ is a defined symbol: it evaluates to either $\top$ or $\bot$. Instead, the other operators $\Omega = \{0, 1, +, \top, \bot\}$ are constructor symbols.

A constructor variant is variant that is a constructor term. For example, $\top$ and $\bot$ are constructor variants of $x > y$, but $x > y$ is not.

Constructor Variants and Constructor Unifiers

In $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ the predicate $>$ is a defined symbol: it evaluates to either $\top$ or $\bot$. Instead, the other operators $\Omega = \{0, 1, +, \top, \bot\}$ are constructor symbols.

A constructor variant is variant that is a constructor term. For example, $\top$ and $\bot$ are constructor variants of $x > y$, but $x > y$ is not.

A constructor $E \cup B$-unifier of $u \equiv v$

Constructor Variants and Constructor Unifiers

In $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ the predicate $>$ is a defined symbol: it evaluates to either $\top$ or $\bot$. Instead, the other operators $\Omega = \{0, 1, +, \top, \bot\}$ are constructor symbols.

A constructor variant is variant that is a constructor term. For example, $\top$ and $\bot$ are constructor variants of $x > y$, but $x > y$ is not.

A constructor $E \cup B$-unifier of $u \equiv v$ has the form $\gamma\beta$ where $(u' \equiv v', \gamma)$ is a variant of $u \equiv v$ with u', v' constructor terms and β is a B-unifier of $u' \equiv v'$.

Constructor Variants and Constructor Unifiers

In $\mathcal{N}_{+,>} = (\Sigma, E \cup ACU)$ the predicate $>$ is a defined symbol: it evaluates to either $\top$ or $\bot$. Instead, the other operators $\Omega = \{0, 1, +, \top, \bot\}$ are constructor symbols.

A constructor variant is variant that is a constructor term. For example, $\top$ and $\bot$ are constructor variants of $x > y$, but $x > y$ is not.

A constructor $E \cup B$-unifier of $u \equiv v$ has the form $\gamma\beta$ where $(u' \equiv v', \gamma)$ is a variant of $u \equiv v$ with u', v' constructor terms and β is a B-unifier of $u' \equiv v'$.

For example, $\{x \mapsto y\}$ is not a constructor unifier of $x > z \equiv y > z$.

OS-Compact Theories

An equational order-sorted theory (Ω, E_Ω) is OS-compact iff:

OS-Compact Theories

An equational order-sorted theory (Ω, E_Ω) is OS-compact iff:

1. E_Ω-unification is finitary, and

OS-Compact Theories

An equational order-sorted theory (Ω, E_Ω) is OS-compact iff:

1. E_Ω-unification is finitary, and
2. a conjunction of disequalities $\bigwedge_{1 \leq i \leq n} u_i \neq v_i$ where all variables have infinite sorts is satisfiable in T_{Ω/E_Ω} iff $u_i \neq_{E_\Omega} v_i$, $1 \leq i \leq n$.

OS-Compact Theories

An equational order-sorted theory (Ω, E_Ω) is OS-compact iff:

1. E_Ω-unification is finitary, and
2. a conjunction of disequalities $\bigwedge_{1 \leq i \leq n} u_i \neq v_i$ where all variables have infinite sorts is satisfiable in T_{Ω/E_Ω} iff $u_i \neq_{E_\Omega} v_i$, $1 \leq i \leq n$.

Theorem. If (Ω, E_Ω) is OS-compact, then satisfiability of QF Ω-formulas in T_{Ω/E_Ω} is decidable.

OS-Compact Theories

An equational order-sorted theory (Ω, E_Ω) is OS-compact iff:

1. E_Ω-unification is finitary, and
2. a conjunction of disequalities $\bigwedge_{1 \leq i \leq n} u_i \neq v_i$ where all variables have infinite sorts is satisfiable in T_{Ω/E_Ω} iff $u_i \neq_{E_\Omega} v_i$, $1 \leq i \leq n$.

Theorem. If (Ω, E_Ω) is OS-compact, then satisfiability of QF Ω-formulas in T_{Ω/E_Ω} is decidable.

Remark. The notion of OS-compact theory and the above theorem generalize a similar notion and theorem by H. Comon.

OS-Compact Theories

An equational order-sorted theory (Ω, E_Ω) is OS-compact iff:

1. E_Ω-unification is finitary, and
2. a conjunction of disequalities $\bigwedge_{1 \leq i \leq n} u_i \neq v_i$ where all variables have infinite sorts is satisfiable in T_{Ω/E_Ω} iff $u_i \neq_{E_\Omega} v_i$, $1 \leq i \leq n$.

Theorem. If (Ω, E_Ω) is OS-compact, then satisfiability of QF Ω-formulas in T_{Ω/E_Ω} is decidable.

Remark. The notion of OS-compact theory and the above theorem generalize a similar notion and theorem by H. Comon.

Theorem. (Ω, B) is OS-compact for any Ω with B any combination of associativity and/or commutativity and/or identity axioms, except associativity without commutativity.

Main Theorem Let $(\Sigma, E \cup B)$ be FVP with B having a finitary unification algorithm,

Main Theorem Let $(\Sigma, E \cup B)$ be FVP with B having a finitary unification algorithm, and such that (Ω, E_Ω) specifies the Ω-reduct algebra of constructors of $T_{\Sigma/E \cup B}$

Variant-Based Satisfiability

Main Theorem Let $(\Sigma, E \cup B)$ be FVP with B having a finitary unification algorithm, and such that (Ω, E_Ω) specifies the Ω-reduct algebra of constructors of $T_{\Sigma/E\cup B}$ (i.e., $T_{\Sigma/E\cup B}|_\Omega \cong T_{\Omega/E_\Omega}$)

Variant-Based Satisfiability

Main Theorem Let $(\Sigma, E \cup B)$ be FVP with B having a finitary unification algorithm, and such that (Ω, E_Ω) specifies the Ω-reduct algebra of constructors of $T_{\Sigma/E \cup B}$ (i.e., $T_{\Sigma/E \cup B}|_\Omega \cong T_{\Omega/E_\Omega}$) and is OS-compact.

Main Theorem Let $(\Sigma, E \cup B)$ be FVP with B having a finitary unification algorithm, and such that (Ω, E_Ω) specifies the Ω-reduct algebra of constructors of $T_{\Sigma/E\cup B}$ (i.e., $T_{\Sigma/E\cup B}|_\Omega \cong T_{\Omega/E_\Omega}$) and is OS-compact.

Then satisfiability of QF Σ-formulas in $T_{\Sigma/E\cup B}$ is decidable.

Main Theorem Let $(\Sigma, E \cup B)$ be FVP with B having a finitary unification algorithm, and such that (Ω, E_Ω) specifies the Ω-reduct algebra of constructors of $T_{\Sigma/E \cup B}$ (i.e., $T_{\Sigma/E\cup B}|_\Omega \cong T_{\Omega/E_\Omega}$) and is OS-compact.

Then satisfiability of QF Σ-formulas in $T_{\Sigma/E \cup B}$ is decidable.

Algorithm: Given conjunction of literals $\bigwedge G \wedge \bigwedge D$,

Variant-Based Satisfiability

Main Theorem Let $(\Sigma, E \cup B)$ be FVP with B having a finitary unification algorithm, and such that (Ω, E_Ω) specifies the Ω-reduct algebra of constructors of $T_{\Sigma/E \cup B}$ (i.e., $T_{\Sigma/E \cup B}|_\Omega \cong T_{\Omega/E_\Omega}$) and is OS-compact.

Then satisfiability of QF Σ-formulas in $T_{\Sigma/E \cup B}$ is decidable.

Algorithm: Given conjunction of literals $\bigwedge G \wedge \bigwedge D$, with G equalities and D disequalities:

Main Theorem Let $(\Sigma, E \cup B)$ be FVP with B having a finitary unification algorithm, and such that (Ω, E_Ω) specifies the Ω-reduct algebra of constructors of $T_{\Sigma/E\cup B}$ (i.e., $T_{\Sigma/E\cup B}|_\Omega \cong T_{\Omega/E_\Omega}$) and is OS-compact.

Then satisfiability of QF Σ-formulas in $T_{\Sigma/E\cup B}$ is decidable.

Algorithm: Given conjunction of literals $\bigwedge G \wedge \bigwedge D$, with G equalities and D disequalities:

1. compute constructor variant $E \cup B$-unifiers α of $\bigwedge G$,

Main Theorem Let $(\Sigma, E \cup B)$ be FVP with B having a finitary unification algorithm, and such that (Ω, E_Ω) specifies the Ω-reduct algebra of constructors of $T_{\Sigma/E \cup B}$ (i.e., $T_{\Sigma/E \cup B}|_\Omega \cong T_{\Omega/E_\Omega}$) and is OS-compact.

Then satisfiability of QF Σ-formulas in $T_{\Sigma/E \cup B}$ is decidable.

Algorithm: Given conjunction of literals $\bigwedge G \wedge \bigwedge D$, with G equalities and D disequalities:

1. compute constructor variant $E \cup B$-unifiers α of $\bigwedge G$,
2. compute the constructor E, B-variants $\bigwedge D'$ of $\bigwedge D\alpha$, and

Variant-Based Satisfiability

Main Theorem Let $(\Sigma, E \cup B)$ be FVP with B having a finitary unification algorithm, and such that (Ω, E_Ω) specifies the Ω-reduct algebra of constructors of $T_{\Sigma/E\cup B}$ (i.e., $T_{\Sigma/E\cup B}|_\Omega \cong T_{\Omega/E_\Omega}$) and is OS-compact.

Then satisfiability of QF Σ-formulas in $T_{\Sigma/E\cup B}$ is decidable.

Algorithm: Given conjunction of literals $\bigwedge G \wedge \bigwedge D$, with G equalities and D disequalities:

1. compute constructor variant $E \cup B$-unifiers α of $\bigwedge G$,
2. compute the constructor E, B-variants $\bigwedge D'$ of $\bigwedge D\alpha$, and
3. for each $u' \neq v'$ in $\bigwedge D'$ check that $u' \neq_{E_\Omega} v'$.

Example of Variant-Based Satisfiability

Consider the quantifier-free formula:

Example of Variant-Based Satisfiability

Consider the quantifier-free formula:

$$head(l) > head(l') = \top \wedge head(l) > 1{+}1{+}1 = \bot \wedge \{(1{+}1); nil\} \subseteq \{l, l', \emptyset\} \neq tt$$

Example of Variant-Based Satisfiability

Consider the quantifier-free formula:

$$head(l) > head(l') = \top \wedge head(l) > 1+1+1 = \bot \wedge \{(1+1); nil\} \subseteq \{l, l', \emptyset\} \neq tt$$

in the composition of:

Example of Variant-Based Satisfiability

Consider the quantifier-free formula:

$$head(l) > head(l') = \top \wedge head(l) > 1{+}1{+}1 = \bot \wedge \{(1{+}1); nil\} \subseteq \{l, l', \emptyset\} \neq tt$$

in the composition of: (i) Presburger arithmetic $\mathcal{N}_{+,>}$, (ii)

Example of Variant-Based Satisfiability

Consider the quantifier-free formula:

$$head(l) > head(l') = \top \wedge head(l) > 1{+}1{+}1 = \bot \wedge \{(1{+}1); nil\} \subseteq \{l, l', \emptyset\} \neq tt$$

in the composition of: (i) Presburger arithmetic $\mathcal{N}_{+,>}$, (ii) the parameterized theory of lists $\mathcal{L}[X]$, and

Example of Variant-Based Satisfiability

Consider the quantifier-free formula:

$$head(l) > head(l') = \top \wedge head(l) > 1{+}1{+}1 = \bot \wedge \{(1{+}1); nil\} \subseteq \{l, l', \emptyset\} \neq tt$$

in the composition of: (i) Presburger arithmetic $\mathcal{N}_{+,>}$, (ii) the parameterized theory of lists $\mathcal{L}[X]$, and and (iii) the parameterized theory of hereditarily finite sets $\mathcal{H}[Y]$.

Example of Variant-Based Satisfiability

Consider the quantifier-free formula:

$$head(l) > head(l') = \top \wedge head(l) > 1+1+1 = \bot \wedge \{(1+1); nil\} \subseteq \{l, l', \emptyset\} \neq tt$$

in the composition of: (i) Presburger arithmetic $\mathcal{N}_{+,>}$, (ii) the parameterized theory of lists $\mathcal{L}[X]$, and and (iii) the parameterized theory of hereditarily finite sets $\mathcal{H}[Y]$. These three theories and their composition are FVP and have decidable satisfiability.

Consider the quantifier-free formula:

$$head(l) > head(l') = \top \wedge head(l) > 1{+}1{+}1 = \bot \wedge \{(1{+}1); nil\} \subseteq \{l, l', \emptyset\} \neq tt$$

in the composition of: (i) Presburger arithmetic $\mathcal{N}_{+,>}$, (ii) the parameterized theory of lists $\mathcal{L}[X]$, and and (iii) the parameterized theory of hereditarily finite sets $\mathcal{H}[Y]$. These three theories and their composition are FVP and have decidable satisfiability. To decide satisfiability we:

Example of Variant-Based Satisfiability

Consider the quantifier-free formula:

$$head(l) > head(l') = \top \wedge head(l) > 1{+}1{+}1 = \bot \wedge \{(1{+}1); nil\} \subseteq \{l, l', \emptyset\} \neq tt$$

in the composition of: (i) Presburger arithmetic $\mathcal{N}_{+,>}$, (ii) the parameterized theory of lists $\mathcal{L}[X]$, and and (iii) the parameterized theory of hereditarily finite sets $\mathcal{H}[Y]$. These three theories and their composition are FVP and have decidable satisfiability. To decide satisfiability we:

1. first solve the sytem of equations $head(l) > head(l') = \top \wedge head(l) > 1 + 1 + 1 = \bot$ modulo the composed theory. There are six constructor unifiers. The first is: $\alpha = \{l \mapsto (1 + 1 + 1); l_1, l' \mapsto (1 + 1); l_2\}$.

Example of Variant-Based Satisfiability

Consider the quantifier-free formula:

$$head(l) > head(l') = \top \wedge head(l) > 1{+}1{+}1 = \bot \wedge \{(1{+}1); nil\} \subseteq \{l, l', \emptyset\} \neq tt$$

in the composition of: (i) Presburger arithmetic $\mathcal{N}_{+,>}$, (ii) the parameterized theory of lists $\mathcal{L}[X]$, and and (iii) the parameterized theory of hereditarily finite sets $\mathcal{H}[Y]$. These three theories and their composition are FVP and have decidable satisfiability. To decide satisfiability we:

1. first solve the sytem of equations $head(l) > head(l') = \top \wedge head(l) > 1 + 1 + 1 = \bot$ modulo the composed theory. There are six constructor unifiers. The first is: $\alpha = \{l \mapsto (1 + 1 + 1); l_1, l' \mapsto (1 + 1); l_2\}$.
2. This shows that the formula is satisfiable, because $\{(1 + 1); nil\} \subseteq \{(1 + 1 + 1); l_1, (1 + 1); l_2, \emptyset\} \neq tt$, is irreducible by the equations for $\subseteq$ modulo ACU.

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

1. HF sets do not seem to be supported by any of the SMT solvers in the Wikipedia SMT solver page,

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

1. HF sets do not seem to be supported by any of the SMT solvers in the Wikipedia SMT solver page, yet HF sets and the three theories are easily definable by rewrite rules.

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

1. HF sets do not seem to be supported by any of the SMT solvers in the Wikipedia SMT solver page, yet HF sets and the three theories are easily definable by rewrite rules.
2. Even if Presburger arithmetic, lists, and HF sets were available in a standard SMT solver, a Nelson-Oppen (NO) combination procedure would have been needed;

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

1. HF sets do not seem to be supported by any of the SMT solvers in the Wikipedia SMT solver page, yet HF sets and the three theories are easily definable by rewrite rules.
2. Even if Presburger arithmetic, lists, and HF sets were available in a standard SMT solver, a Nelson-Oppen (NO) combination procedure would have been needed; here we just take the union of the three theories:

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

1. HF sets do not seem to be supported by any of the SMT solvers in the Wikipedia SMT solver page, yet HF sets and the three theories are easily definable by rewrite rules.
2. Even if Presburger arithmetic, lists, and HF sets were available in a standard SMT solver, a Nelson-Oppen (NO) combination procedure would have been needed; here we just take the union of the three theories: no NO combination is needed.

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

1. HF sets do not seem to be supported by any of the SMT solvers in the Wikipedia SMT solver page, yet HF sets and the three theories are easily definable by rewrite rules.
2. Even if Presburger arithmetic, lists, and HF sets were available in a standard SMT solver, a Nelson-Oppen (NO) combination procedure would have been needed; here we just take the union of the three theories: no NO combination is needed.

Many other theories can be made decidable this way, including:

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

1. HF sets do not seem to be supported by any of the SMT solvers in the Wikipedia SMT solver page, yet HF sets and the three theories are easily definable by rewrite rules.
2. Even if Presburger arithmetic, lists, and HF sets were available in a standard SMT solver, a Nelson-Oppen (NO) combination procedure would have been needed; here we just take the union of the three theories: no NO combination is needed.

Many other theories can be made decidable this way, including: (i) any FVP theory whose constructor subspecification is OS-compact;

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

1. HF sets do not seem to be supported by any of the SMT solvers in the Wikipedia SMT solver page, yet HF sets and the three theories are easily definable by rewrite rules.
2. Even if Presburger arithmetic, lists, and HF sets were available in a standard SMT solver, a Nelson-Oppen (NO) combination procedure would have been needed; here we just take the union of the three theories: no NO combination is needed.

Many other theories can be made decidable this way, including: (i) any FVP theory whose constructor subspecification is OS-compact; (ii) all constructor-selector parameterized data types;

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

1. HF sets do not seem to be supported by any of the SMT solvers in the Wikipedia SMT solver page, yet HF sets and the three theories are easily definable by rewrite rules.
2. Even if Presburger arithmetic, lists, and HF sets were available in a standard SMT solver, a Nelson-Oppen (NO) combination procedure would have been needed; here we just take the union of the three theories: no NO combination is needed.

Many other theories can be made decidable this way, including: (i) any FVP theory whose constructor subspecification is OS-compact; (ii) all constructor-selector parameterized data types; (iii) sets, multisets and HF sets parameterized types; and

Example of Variant-Based Satisfiability II

Although this is a simple example, it illustrates the extensible nature of variant-based satisfiability because:

1. HF sets do not seem to be supported by any of the SMT solvers in the Wikipedia SMT solver page, yet HF sets and the three theories are easily definable by rewrite rules.
2. Even if Presburger arithmetic, lists, and HF sets were available in a standard SMT solver, a Nelson-Oppen (NO) combination procedure would have been needed; here we just take the union of the three theories: no NO combination is needed.

Many other theories can be made decidable this way, including: (i) any FVP theory whose constructor subspecification is OS-compact; (ii) all constructor-selector parameterized data types; (iii) sets, multisets and HF sets parameterized types; and (iv) various numeric functions on the naturals and integers.

Symbolic Reachability Analysis in a Nutshell

We can symbolically analyze the reachability properties of a concurrent system

Symbolic Reachability Analysis in a Nutshell

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by:

Symbolic Reachability Analysis in a Nutshell

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by: (i) representing sets of states as terms with variables, and

Symbolic Reachability Analysis in a Nutshell

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by: (i) representing sets of states as terms with variables, and (ii) performing narrowing with rules R modulo $E \cup B$,

Symbolic Reachability Analysis in a Nutshell

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by: (i) representing sets of states as terms with variables, and (ii) performing narrowing with rules R modulo $E \cup B$, where the narrowing relation $t \rightsquigarrow_{R/E \cup B} t'$

Symbolic Reachability Analysis in a Nutshell

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by: (i) representing sets of states as terms with variables, and (ii) performing narrowing with rules R modulo $E \cup B$, where the narrowing relation $t \leadsto_{R/E \cup B} t'$ is defined iff there is:

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by: (i) representing sets of states as terms with variables, and (ii) performing narrowing with rules R modulo $E \cup B$, where the narrowing relation $t \rightsquigarrow_{R/E \cup B} t'$ is defined iff there is:

- a rule $l \to r$ in R; and

Symbolic Reachability Analysis in a Nutshell

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by: (i) representing sets of states as terms with variables, and (ii) performing narrowing with rules R modulo $E \cup B$, where the narrowing relation $t \leadsto_{R/E \cup B} t'$ is defined iff there is:

- a rule $l \to r$ in R; and
- a $E \cup B$-variant unifier σ

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by: (i) representing sets of states as terms with variables, and (ii) performing narrowing with rules R modulo $E \cup B$, where the narrowing relation $t \leadsto_{R/E \cup B} t'$ is defined iff there is:

- a rule $l \to r$ in R; and
- a $E \cup B$-variant unifier σ such that $\sigma(t) =_{(E \cup B)} \sigma(l)$, and $t' = \sigma(r)$.

Symbolic Reachability Analysis in a Nutshell

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by: (i) representing sets of states as terms with variables, and (ii) performing narrowing with rules R modulo $E \cup B$, where the narrowing relation $t \rightsquigarrow_{R/E \cup B} t'$ is defined iff there is:

- a rule $l \rightarrow r$ in R; and
- a $E \cup B$-variant unifier σ such that $\sigma(t) =_{(E \cup B)} \sigma(l)$, and $t' = \sigma(r)$.

This method is complete for reachability analysis: an instance of the states described by t can reach an instance of those described by t' in the system specified by $\mathcal{R}$ iff

Symbolic Reachability Analysis in a Nutshell

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by: (i) representing sets of states as terms with variables, and (ii) performing narrowing with rules R modulo $E \cup B$, where the narrowing relation $t \leadsto_{R/E \cup B} t'$ is defined iff there is:

- a rule $l \to r$ in R; and
- a $E \cup B$-variant unifier σ such that $\sigma(t) =_{(E \cup B)} \sigma(l)$, and $t' = \sigma(r)$.

This method is complete for reachability analysis: an instance of the states described by t can reach an instance of those described by t' in the system specified by $\mathcal{R}$ iff $t \leadsto_{R/E \cup B} t'$.

Symbolic Reachability Analysis in a Nutshell

We can symbolically analyze the reachability properties of a concurrent system specified by a topmost rewrite theory $\mathcal{R} = (\Sigma, E \cup B, R)$ with $E \cup B$ FVP by: (i) representing sets of states as terms with variables, and (ii) performing narrowing with rules R modulo $E \cup B$, where the narrowing relation $t \rightsquigarrow_{R/E \cup B} t'$ is defined iff there is:

- a rule $l \rightarrow r$ in R; and
- a $E \cup B$-variant unifier σ such that $\sigma(t) =_{(E \cup B)} \sigma(l)$, and $t' = \sigma(r)$.

This method is complete for reachability analysis: an instance of the states described by t can reach an instance of those described by t' in the system specified by $\mathcal{R}$ iff $t \rightsquigarrow_{R/E \cup B} t'$.

Note that narrowing happens at two levels:

- with rules R modulo $E \cup B$ to perform symbolic transitions
- with E modulo B to compute $E \cup B$-unifiers.

Maude can performs symbolic reachability analysis as just described.

Maude can performs symbolic reachability analysis as just described. The Maude-NPA tool of Escobar, Meadows and Meseguer, analyzes crypto protocols modeled as rewrite theories $\mathcal{P} = (\Sigma, E \cup B, R)$ by narrowing with rules R modulo FVP equations $E \cup B$.

Maude and the Maude-NPA Crypto Protocol Analyzer

Maude can performs symbolic reachability analysis as just described. The Maude-NPA tool of Escobar, Meadows and Meseguer, analyzes crypto protocols modeled as rewrite theories $\mathcal{P} = (\Sigma, E \cup B, R)$ by narrowing with rules R modulo FVP equations $E \cup B$.

Many protocols have been analyzed modulo non-trivial theories such as: (i) encryption-decryption; (ii) exclusive or; (iii) Diffie-Hellman exponentiation; (iv) homomorphic encryption, and combinations of such theories.

Maude can performs symbolic reachability analysis as just described. The Maude-NPA tool of Escobar, Meadows and Meseguer, analyzes crypto protocols modeled as rewrite theories $\mathcal{P} = (\Sigma, E \cup B, R)$ by narrowing with rules R modulo FVP equations $E \cup B$.

Many protocols have been analyzed modulo non-trivial theories such as: (i) encryption-decryption; (ii) exclusive or; (iii) Diffie-Hellman exponentiation; (iv) homomorphic encryption, and combinations of such theories.

Although Maude-NPA deals with unbounded sessions for which reachability is undecidable, its use of very effective symbolic state space reduction techiques often makes the state space finite, allowing full verification.

Maude and the Maude-NPA Crypto Protocol Analyzer

Maude can performs symbolic reachability analysis as just described. The Maude-NPA tool of Escobar, Meadows and Meseguer, analyzes crypto protocols modeled as rewrite theories $\mathcal{P} = (\Sigma, E \cup B, R)$ by narrowing with rules R modulo FVP equations $E \cup B$.

Many protocols have been analyzed modulo non-trivial theories such as: (i) encryption-decryption; (ii) exclusive or; (iii) Diffie-Hellman exponentiation; (iv) homomorphic encryption, and combinations of such theories.

Although Maude-NPA deals with unbounded sessions for which reachability is undecidable, its use of very effective symbolic state space reduction techiques often makes the state space finite, allowing full verification.

Both tool are available at `http://maude.cs.illinois.edu`

The Maude-NPA Crypto Protocol Analyzer (II)

Homomorphic encryption is challenging: the theories H and AGH are not FVP, and combining their unification algorithms with those of other theories is computationally expensive.

The Maude-NPA Crypto Protocol Analyzer (II)

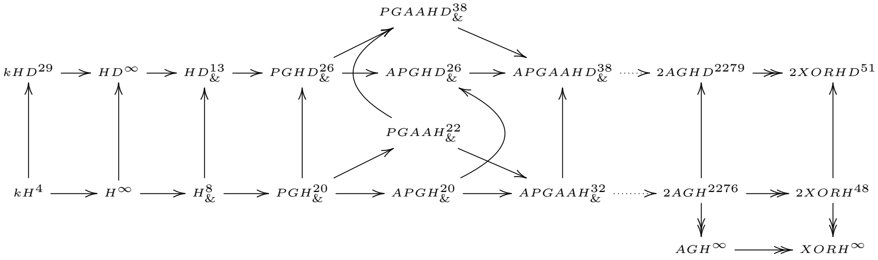
Homomorphic encryption is challenging: the theories H and AGH are not FVP, and combining their unification algorithms with those of other theories is computationally expensive.

In joint work with Yang et al., several FVP theories of homomorphic encryption have been used with protocols in Maude-NPA by trading accuracy and variant complexity.

The Maude-NPA Crypto Protocol Analyzer (II)

Homomorphic encryption is challenging: the theories *H* and *AGH* are not FVP, and combining their unification algorithms with those of other theories is computationally expensive.

In joint work with Yang et al., several FVP theories of homomorphic encryption have been used with protocols in Maude-NPA by trading accuracy and variant complexity.



Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as:

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**,

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**,

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and Narrowing-based **Symbolic Reachability Analysis.**

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and Narrowing-based **Symbolic Reachability Analysis.** Theory-generic methods can make formal methods much more extensible.

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and Narrowing-based **Symbolic Reachability Analysis.**Theory-generic methods can make formal methods much more extensible. For example, other formal tools like:

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and Narrowing-based **Symbolic Reachability Analysis.** Theory-generic methods can make formal methods much more extensible. For example, other formal tools like:

1. **Maude's Symbolic LTL Model Checker** of Bae et al.

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and Narrowing-based **Symbolic Reachability Analysis.** Theory-generic methods can make formal methods much more extensible. For example, other formal tools like:

1. **Maude's Symbolic LTL Model Checker** of Bae et al.
2. The **Tamarin** security prover at ETH

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and Narrowing-based **Symbolic Reachability Analysis.**Theory-generic methods can make formal methods much more extensible. For example, other formal tools like:

1. **Maude's Symbolic LTL Model Checker** of Bae et al.
2. The **Tamarin** security prover at ETH
3. The **Reachability Logic Theorem Prover** of S. Skeirik, A. Stefanescu, and J. Meseguer, and

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and Narrowing-based **Symbolic Reachability Analysis.**Theory-generic methods can make formal methods much more extensible. For example, other formal tools like:

1. **Maude's Symbolic LTL Model Checker** of Bae et al.
2. The **Tamarin** security prover at ETH
3. The **Reachability Logic Theorem Prover** of S. Skeirik, A. Stefanescu, and J. Meseguer, and
4. The **Partial Evaluation Framework** for Maude functional modules of Alpuente et al. at TU of Valencia.

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and Narrowing-based **Symbolic Reachability Analysis.**Theory-generic methods can make formal methods much more extensible. For example, other formal tools like:

1. **Maude's Symbolic LTL Model Checker** of Bae et al.
2. The **Tamarin** security prover at ETH
3. The **Reachability Logic Theorem Prover** of S. Skeirik, A. Stefanescu, and J. Meseguer, and
4. The **Partial Evaluation Framework** for Maude functional modules of Alpuente et al. at TU of Valencia.

All these tools use the above-mentioned theory-generic symbolic methods provided by Maude.

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and Narrowing-based **Symbolic Reachability Analysis.**Theory-generic methods can make formal methods much more extensible. For example, other formal tools like:

1. **Maude's Symbolic LTL Model Checker** of Bae et al.
2. The **Tamarin** security prover at ETH
3. The **Reachability Logic Theorem Prover** of S. Skeirik, A. Stefanescu, and J. Meseguer, and
4. The **Partial Evaluation Framework** for Maude functional modules of Alpuente et al. at TU of Valencia.

All these tools use the above-mentioned theory-generic symbolic methods provided by Maude. In these and other ways, Maude becomes not just a declarative language,

Conclusion: Towards Extensible Formal Methods

I have emphasized the importance of theory-generic symbolic methods such as: **Variants**, **Variant Unification**, **Variant-Based Satisfiability**, and Narrowing-based **Symbolic Reachability Analysis.** Theory-generic methods can make formal methods much more extensible. For example, other formal tools like:

1. **Maude's Symbolic LTL Model Checker** of Bae et al.
2. The **Tamarin** security prover at ETH
3. The **Reachability Logic Theorem Prover** of S. Skeirik, A. Stefanescu, and J. Meseguer, and
4. The **Partial Evaluation Framework** for Maude functional modules of Alpuente et al. at TU of Valencia.

All these tools use the above-mentioned theory-generic symbolic methods provided by Maude. In these and other ways, Maude becomes not just a declarative language, but a formal framework for programming and verification.

Acknowledgements

The work on:

Acknowledgements

The work on:

1. Symbolic Methods in Maude is joint work of the Maude Team, and owes much to Steven Eker's high-performance implementations.

Acknowledgements

The work on:

1. Symbolic Methods in Maude is joint work of the Maude Team, and owes much to Steven Eker's high-performance implementations.
2. Folding Variant Narrowing is joint with S. Escobar and R. Sasse;

Acknowledgements

The work on:

1. Symbolic Methods in Maude is joint work of the Maude Team, and owes much to Steven Eker's high-performance implementations.
2. Folding Variant Narrowing is joint with S. Escobar and R. Sasse;
3. Variant-Based satisfiability is joint with S. Skeirik and R. Gutiérrez;

Acknowledgements

The work on:

1. Symbolic Methods in Maude is joint work of the Maude Team, and owes much to Steven Eker's high-performance implementations.
2. Folding Variant Narrowing is joint with S. Escobar and R. Sasse;
3. Variant-Based satisfiability is joint with S. Skeirik and R. Gutiérrez;
4. Maude-NPA is joint with C. Meadows, S. Escobar, and Ph.D. students at Illinois, Valencia, and Oslo;

Acknowledgements

The work on:

1. Symbolic Methods in Maude is joint work of the Maude Team, and owes much to Steven Eker's high-performance implementations.
2. Folding Variant Narrowing is joint with S. Escobar and R. Sasse;
3. Variant-Based satisfiability is joint with S. Skeirik and R. Gutiérrez;
4. Maude-NPA is joint with C. Meadows, S. Escobar, and Ph.D. students at Illinois, Valencia, and Oslo;
5. Maude's Symbolic LTL Model Checker is joint with K. Bae and S. Escobar;

Acknowledgements

The work on:

1. Symbolic Methods in Maude is joint work of the Maude Team, and owes much to Steven Eker's high-performance implementations.
2. Folding Variant Narrowing is joint with S. Escobar and R. Sasse;
3. Variant-Based satisfiability is joint with S. Skeirik and R. Gutiérrez;
4. Maude-NPA is joint with C. Meadows, S. Escobar, and Ph.D. students at Illinois, Valencia, and Oslo;
5. Maude's Symbolic LTL Model Checker is joint with K. Bae and S. Escobar;
6. Maude's Partial Evaluation Framework is joint with M. Alpuente, A. Cuenca-Ortega, and S. Escobar at TU Valencia.

Thank you!