Satisfiability Checking

First-Order Logic

Prof. Dr. Erika Ábrahám

RWTH Aachen University
Informatik 2
LuFG Theory of Hybrid Systems

WS 19/20

First-order logic

- We have seen that natural languages are not well-suited for correct reasoning.
- Propositional logic is useful but sometimes not expressive enough for modeling.

First-order logic

- We have seen that natural languages are not well-suited for correct reasoning.
- Propositional logic is useful but sometimes not expressive enough for modeling.

First-order (FO) logic is a framework with the syntactical ingredients:

1. Theory symbols: constants, variables, function symbols
2. Lifting from theory to the logical level: predicate symbols
3. Logical symbols: Logical connectives and quantifiers

First-order logic

- We have seen that natural languages are not well-suited for correct reasoning.
- Propositional logic is useful but sometimes not expressive enough for modeling.

First-order (FO) logic is a framework with the syntactical ingredients:

1. Theory symbols: constants, variables, function symbols
2. Lifting from theory to the logical level: predicate symbols
3. Logical symbols: Logical connectives and quantifiers

- 3 is fixed
- Fixing 1 and 2 gives different FO instances

Constants, variables, function symbols, terms

Theory symbols: constants, variables, function symbols

Constants, variables, function symbols, terms

Theory symbols: constants, variables, function symbols

Example:

Constants:	$0, 1$
Variables:	$x, y, z, \ldots$
Function symbol	binary $+$

Constants, variables, function symbols, terms

Theory symbols: constants, variables, function symbols

Example:

Constants:	$0, 1$
Variables:	$x, y, z, \ldots$
Function symbol	binary $+$

Terms (theory expressions) are inductively defined by the following rules:

1. All constants and variables are terms.
2. If $t_1, \ldots, t_n$ ($n > 0$) are terms and f an n-ary function symbol then $f(t_1, ..., t_n)$ is a term.

Only strings obtained by finitely many applications of these rules are terms.

Constants, variables, function symbols, terms

Theory symbols: constants, variables, function symbols

Example:

Constants:	$0, 1$
Variables:	$x, y, z, \dots$
Function symbol	binary $+$

Terms (theory expressions) are inductively defined by the following rules:

1. All constants and variables are terms.
2. If $t_1, \dots, t_n$ ($n > 0$) are terms and f an n-ary function symbol then $f(t_1, ..., t_n)$ is a term.

Only strings obtained by finitely many applications of these rules are terms.

Example terms: 0, x, $+(0, 1)$, $+(x, 1)$, $+(x, +(y, 1))$

Constants, variables, function symbols, terms

Theory symbols: constants, variables, function symbols

Example:

Constants:	$0, 1$
Variables:	$x, y, z, \ldots$
Function symbol	binary $+$

Terms (theory expressions) are inductively defined by the following rules:

1. All constants and variables are terms.
2. If $t_1, \ldots, t_n$ $(n > 0)$ are terms and f an n-ary function symbol then $f(t_1, ..., t_n)$ is a term.

Only strings obtained by finitely many applications of these rules are terms.

Example terms: 0, x, $(0 + 1)$, $(x + 1)$, $(x + (y + 1))$

Predicates, constraints

Predicates lift terms from the theory to the logical level.

Predicates, constraints

Predicates lift terms from the theory to the logical level.

Example predicate symbols: binary $\geq, >, =, <, \leq$

Predicates, constraints

Predicates lift terms from the theory to the logical level.

Example predicate symbols: binary $\geq, >, =, <, \leq$

(Theory) constraints are inductively defined by the following rule:

1. If P is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms then $P(t_1, \ldots, t_n)$ is a constraint.

Only strings obtained by finitely many applications of this rule are constraints.

Predicates, constraints

Predicates lift terms from the theory to the logical level.

Example predicate symbols: binary $\geq, >, =, <, \leq$

(Theory) constraints are inductively defined by the following rule:

1. If P is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms then $P(t_1, \ldots, t_n)$ is a constraint.

Only strings obtained by finitely many applications of this rule are constraints.

Example constraints: $x < (x + 1)$, $((x + 1) + y) = ((x + y) + 1)$

Logical connectives and quantifiers, formulas

- Logical connectives: unary $\neg$, binary $\wedge, \vee, \rightarrow, \leftrightarrow, \ldots$
- Universal quantifier $\forall$ ("for all"), existential quantifier $\exists$ ("exists")

Logical connectives and quantifiers, formulas

- Logical connectives: unary $\neg$, binary $\wedge, \vee, \rightarrow, \leftrightarrow, \ldots$
- Universal quantifier $\forall$ ("for all"), existential quantifier $\exists$ ("exists")

(Well-formed) formulas are inductively defined by the following rules:

1. If c is a constraint then c is a formula (called atomic formula).
2. If φ is a formula then $(\neg\varphi)$ is a formula.
3. If φ and ψ are formulas then $(\varphi \wedge \psi)$ is a formula.
4. Similar rules apply to other binary logical connectives.
5. If φ is a formula and x is a variable, then $(\forall x.\ \varphi)$ and $(\exists x.\ \varphi)$ are formulas.

Only expressions which can be obtained by finitely many applications of these rules are formulas.

Logical connectives and quantifiers, formulas

- Logical connectives: unary $\neg$, binary $\wedge, \vee, \rightarrow, \leftrightarrow, \ldots$
- Universal quantifier $\forall$ (“for all”), existential quantifier $\exists$ (“exists”)

(Well-formed) formulas are inductively defined by the following rules:

1. If c is a constraint then c is a formula (called atomic formula).
2. If φ is a formula then $(\neg\varphi)$ is a formula.
3. If φ and ψ are formulas then $(\varphi \wedge \psi)$ is a formula.
4. Similar rules apply to other binary logical connectives.
5. If φ is a formula and x is a variable, then $(\forall x.\ \varphi)$ and $(\exists x.\ \varphi)$ are formulas.

Only expressions which can be obtained by finitely many applications of these rules are formulas.

Example formulas:

- $x < (x+1)$ (atomic formula)
- $(\neg\ x < 0)$
- $(x < (x+1) \wedge ((x+1)+y) = ((x+y)+1))$
- $\forall x.\exists y.\ y = (x+1)$

Example

Assume the argumentation:

1. All men are mortal.
2. Socrates is a man.
3. Therefore, Socrates is mortal.

Example

Assume the argumentation:

1. All men are mortal.
2. Socrates is a man.
3. Therefore, Socrates is mortal.

We can formalize it by defining

Constants:	*Socrates*
Variables:	x
Predicate symbols:	unary *isMen*, *isMortal*

Example

Assume the argumentation:

1. All men are mortal.
2. Socrates is a man.
3. Therefore, Socrates is mortal.

We can formalize it by defining

Constants:	$\mathit{Socrates}$
Variables:	x
Predicate symbols:	unary isMen, $\mathit{isMortal}$

Formalization:

1. $\forall x.\ \mathit{isMen}(x) \rightarrow \mathit{isMortal}(x)$
2. $\mathit{isMen}(\mathit{Socrates})$
3. $\mathit{isMortal}(\mathit{Socrates})$

Some remarks and notation

- Constants can also be seen as function symbols of arity 0.
- Sometimes equality ($=$) is included as a logical symbol.
- Note: the logical connectives negation ($\neg$) and conjunction ($\wedge$) and the existential quantifier ($\exists$) would be sufficient, the remaining syntax ($\vee, \rightarrow, \leftrightarrow, \ldots, \forall$) are syntactic sugar.

Some remarks and notation

- Constants can also be seen as function symbols of arity 0.
- Sometimes equality ($=$) is included as a logical symbol.
- Note: the logical connectives negation ($\neg$) and conjunction ($\wedge$) and the existential quantifier ($\exists$) would be sufficient, the remaining syntax ($\vee, \rightarrow, \leftrightarrow, \ldots, \forall$) are syntactic sugar.

We omit parentheses whenever we may restore them through operator precedence (with left-to-right binding for several occurrences of the same operator):

binds stronger

$\longleftarrow$

$\neg \quad \wedge \quad \vee \quad \rightarrow \quad \leftrightarrow \quad \exists \quad \forall$

Some remarks and notation

- Constants can also be seen as function symbols of arity 0.
- Sometimes equality ($=$) is included as a logical symbol.
- Note: the logical connectives negation ($\neg$) and conjunction ($\wedge$) and the existential quantifier ($\exists$) would be sufficient, the remaining syntax ($\vee, \rightarrow, \leftrightarrow, \ldots, \forall$) are syntactic sugar.

We omit parentheses whenever we may restore them through operator precedence (with left-to-right binding for several occurrences of the same operator):

binds stronger

$\longleftarrow$

$\neg \quad \wedge \quad \vee \quad \rightarrow \quad \leftrightarrow \quad \exists \quad \forall$

Thus, we write:

$$\neg\neg a \quad \text{for} \quad (\neg(\neg a)),$$
$$\exists a.\ \exists b.\ (a \wedge b \rightarrow P(a,b)) \quad \text{for} \quad \exists a.\ \exists b.\ ((a \wedge b) \rightarrow P(a,b))$$

Free and bound variables

Free and bound variables

The free and bound variables of a formula are defined inductively:

Free and bound variables

The free and bound variables of a formula are defined inductively:

- If φ is an atomic formula then a variable x is free in φ iff x occurs in φ. Moreover, there are no bound variables in any atomic formula.

Free and bound variables

The free and bound variables of a formula are defined inductively:

- If φ is an atomic formula then a variable x is free in φ iff x occurs in φ. Moreover, there are no bound variables in any atomic formula.
- A variable x is free in $(\neg\varphi)$ iff x is free in φ. Moreover, x is bound in $(\neg\varphi)$ iff x is bound in φ.

Free and bound variables

The free and bound variables of a formula are defined inductively:

- If φ is an atomic formula then a variable x is free in φ iff x occurs in φ. Moreover, there are no bound variables in any atomic formula.
- A variable x is free in $(\neg\varphi)$ iff x is free in φ. Moreover, x is bound in $(\neg\varphi)$ iff x is bound in φ.
- x is free in $(\varphi \wedge \psi)$ iff x is free in either φ or ψ. Moreover, x is bound in $(\varphi \wedge \psi)$ iff x is bound in either φ or ψ.

Free and bound variables

The free and bound variables of a formula are defined inductively:

- If φ is an atomic formula then a variable x is free in φ iff x occurs in φ. Moreover, there are no bound variables in any atomic formula.
- A variable x is free in $(\neg\varphi)$ iff x is free in φ. Moreover, x is bound in $(\neg\varphi)$ iff x is bound in φ.
- x is free in $(\varphi \wedge \psi)$ iff x is free in either φ or ψ. Moreover, x is bound in $(\varphi \wedge \psi)$ iff x is bound in either φ or ψ.
- The same rule applies to any other binary connective in place of $\wedge$.

Free and bound variables

The free and bound variables of a formula are defined inductively:

- If φ is an atomic formula then a variable x is free in φ iff x occurs in φ. Moreover, there are no bound variables in any atomic formula.
- A variable x is free in $(\neg\varphi)$ iff x is free in φ. Moreover, x is bound in $(\neg\varphi)$ iff x is bound in φ.
- x is free in $(\varphi \wedge \psi)$ iff x is free in either φ or ψ. Moreover, x is bound in $(\varphi \wedge \psi)$ iff x is bound in either φ or ψ.
- The same rule applies to any other binary connective in place of $\wedge$.
- x is free in $(\exists y.\ \varphi)$ iff x is free in φ and x is a symbol different from y. Moreover, x is bound in $(\exists y.\ \varphi)$ iff x is y or x is bound in φ.

Free and bound variables

The free and bound variables of a formula are defined inductively:

- If φ is an atomic formula then a variable x is free in φ iff x occurs in φ. Moreover, there are no bound variables in any atomic formula.
- A variable x is free in $(\neg\varphi)$ iff x is free in φ. Moreover, x is bound in $(\neg\varphi)$ iff x is bound in φ.
- x is free in $(\varphi \wedge \psi)$ iff x is free in either φ or ψ. Moreover, x is bound in $(\varphi \wedge \psi)$ iff x is bound in either φ or ψ.
- The same rule applies to any other binary connective in place of $\wedge$.
- x is free in $(\exists y.\ \varphi)$ iff x is free in φ and x is a symbol different from y. Moreover, x is bound in $(\exists y.\ \varphi)$ iff x is y or x is bound in φ.
- The same rule holds with $\forall$ in place of $\exists$.

Free and bound variables

Examples:

- In $P(z) \vee \forall x.\ \forall y.\ (P(x) \rightarrow Q(z))$, x and y are bound variables, z is a free variable, and w is neither bound nor free.
- In $Q(z) \vee \forall z.P(z)$, z is both bound and free.

Being free or bound is for specific occurrences of variables in a formula.

- In $Q(z) \vee \forall z.P(z)$, the first occurrence of z is free while the second is bound.

Signature Σ, Σ-formula, Σ-sentence

- A signature Σ fixes the set of non-logical symbols.
- A Σ-formula is a formula with non-logical symbols from Σ.
- A Σ-sentence is a Σ-formula without free variables.

Signature Σ, Σ-formula, Σ-sentence

- A signature Σ fixes the set of non-logical symbols.
- A Σ-formula is a formula with non-logical symbols from Σ.
- A Σ-sentence is a Σ-formula without free variables.

In the previous example: $\Sigma = (\mathit{Socrates}, \mathit{isMen}(\cdot), \mathit{isMortal}(\cdot))$ with

- *Socrates* a constant and
- *isMen* and *isMortal* unary predicate symbols.

Signature Σ, Σ-formula, Σ-sentence

- A signature Σ fixes the set of non-logical symbols.
- A Σ-formula is a formula with non-logical symbols from Σ.
- A Σ-sentence is a Σ-formula without free variables.

In the previous example: $\Sigma = (\mathit{Socrates}, \mathit{isMen}(\cdot), \mathit{isMortal}(\cdot))$ with

- *Socrates* a constant and
- *isMen* and *isMortal* unary predicate symbols.

The formulas

1. $\forall x.\ \mathit{isMen}(x) \rightarrow \mathit{isMortal}(x)$
2. $\mathit{isMen}(\mathit{Socrates})$
3. $\mathit{isMortal}(\mathit{Socrates})$

are Σ-sentences (the only variable x is bound).

Further examples

- $\Sigma = \{0, 1, +, >\}$
  - $0, 1$ are constant symbols
  - $+$ is a binary function symbol
  - $>$ is a binary predicate symbol

Further examples

- $\Sigma = \{0, 1, +, >\}$
  - $0, 1$ are constant symbols
  - $+$ is a binary function symbol
  - $>$ is a binary predicate symbol

- Examples of Σ-sentences:

Further examples

- $\Sigma = \{0, 1, +, >\}$
  - $0, 1$ are constant symbols
  - $+$ is a binary function symbol
  - $>$ is a binary predicate symbol

- Examples of Σ-sentences:

  $\exists x.\ \forall y.\ x > y$
  $\forall x.\ \exists y.\ x > y$
  $\forall x.\ x + 1 > x$
  $\forall x.\ \neg(x + 0 > x \lor x > x + 0)$

Further examples

- $\Sigma = \{0, 1, +, *, <, \mathit{isPrime}\}$
  - $0, 1$ constant symbols
  - $+, *$ binary function symbols
  - $<$ binary predicate symbol
  - *isPrime* unary predicate symbol

Further examples

- $\Sigma = \{0, 1, +, *, <, isPrime\}$
  - $0, 1$ constant symbols
  - $+, *$ binary function symbols
  - $<$ binary predicate symbol
  - $isPrime$ unary predicate symbol
- An example Σ-sentence:

$$\forall n.\ (1 < n \rightarrow (\exists p.\ isPrime(p) \land n < p < 2 * n))$$

Example

- Let $\Sigma = \{0, 1, +, =\}$ where $0, 1$ are constants, $+$ is a binary function symbol and $=$ a binary predicate symbol.
- Let $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.

Example

- Let $\Sigma = \{0, 1, +, =\}$ where $0, 1$ are constants, $+$ is a binary function symbol and $=$ a binary predicate symbol.
- Let $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- Q: Is φ true?

Example

- Let $\Sigma = \{0, 1, +, =\}$ where $0, 1$ are constants, $+$ is a binary function symbol and $=$ a binary predicate symbol.
- Let $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- Q: Is φ true?
- A: So far these are only symbols, strings. No meaning yet.

Example

- Let $\Sigma = \{0, 1, +, =\}$ where $0, 1$ are constants, $+$ is a binary function symbol and $=$ a binary predicate symbol.
- Let $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- Q: Is φ true?
- A: So far these are only symbols, strings. No meaning yet.
- Q: What do we need to fix for the semantics?

Example

- Let $\Sigma = \{0, 1, +, =\}$ where $0, 1$ are constants, $+$ is a binary function symbol and $=$ a binary predicate symbol.
- Let $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- Q: Is φ true?
- A: So far these are only symbols, strings. No meaning yet.
- Q: What do we need to fix for the semantics?
- A: We need a domain for the variables. Let's say $\mathbb{N}_0$.

Example

- Let $\Sigma = \{0, 1, +, =\}$ where $0, 1$ are constants, $+$ is a binary function symbol and $=$ a binary predicate symbol.
- Let $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- Q: Is φ true?
- A: So far these are only symbols, strings. No meaning yet.
- Q: What do we need to fix for the semantics?
- A: We need a domain for the variables. Let's say $\mathbb{N}_0$.
- Q: Is φ true in $\mathbb{N}_0$?

Example

- Let $\Sigma = \{0, 1, +, =\}$ where $0, 1$ are constants, $+$ is a binary function symbol and $=$ a binary predicate symbol.
- Let $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- Q: Is φ true?
- A: So far these are only symbols, strings. No meaning yet.
- Q: What do we need to fix for the semantics?
- A: We need a domain for the variables. Let's say $\mathbb{N}_0$.
- Q: Is φ true in $\mathbb{N}_0$?
- A: Depends on the interpretation of '$+$' and '$=$'!

Structures, satisfiability, validity

- A Σ-structure is given by:
  - a domain D,
  - an interpretation I of the non-logical symbols in Σ that maps
    - each constant symbol to a domain element,
    - each function symbol of arity n to a function of type $D^n \rightarrow D$, and
    - each predicate symbol of arity n to a predicate of type $D^n \rightarrow \{0, 1\}$.
- To give meaning to formulas with free variables, we also need an assignment α that maps each (free) variable to a domain element.

Structures, satisfiability, validity

- A Σ-structure is given by:
  - a domain D,
  - an interpretation I of the non-logical symbols in Σ that maps
    - each constant symbol to a domain element,
    - each function symbol of arity n to a function of type $D^n \to D$, and
    - each predicate symbol of arity n to a predicate of type $D^n \to \{0, 1\}$.
- To give meaning to formulas with free variables, we also need an assignment α that maps each (free) variable to a domain element.
- A Σ-formula φ is satisfiable if there exist a Σ-structure S and an assignment α that satisfy it.
  Notation: $S, \alpha \models \varphi$. For Σ-sentences we also write $S \models \varphi$.

Structures, satisfiability, validity

- A Σ-structure is given by:
  - a domain D,
  - an interpretation I of the non-logical symbols in Σ that maps
    - each constant symbol to a domain element,
    - each function symbol of arity n to a function of type $D^n \rightarrow D$, and
    - each predicate symbol of arity n to a predicate of type $D^n \rightarrow \{0, 1\}$.
- To give meaning to formulas with free variables, we also need an assignment α that maps each (free) variable to a domain element.
- A Σ-formula φ is satisfiable if there exist a Σ-structure S and an assignment α that satisfy it.
  Notation: $S, \alpha \models \varphi$. For Σ-sentences we also write $S \models \varphi$.
- A Σ-formula φ is valid if it is satisfied by all Σ-structures and all assignments. Notation: $\models \varphi$.

Semantics

Semantics of terms and formulas under a structure $S = (D, I)$ and an assignment α:

constants:	$[\![c]\!]_{S,\alpha}$	$= I(c)$
variables:	$[\![x]\!]_{S,\alpha}$	$= \alpha(x)$
functions:	$[\![f(t_1, \ldots, t_n)]\!]_{S,\alpha}$	$= I(f)([\![t_1]\!]_{S,\alpha}, \ldots, [\![t_n]\!]_{S,\alpha})$
predicates:	$S, \alpha \models p(t_1, \ldots, t_n)$	iff $I(p)([\![t_1]\!]_{S,\alpha}, \ldots, [\![t_n]\!]_{S,\alpha})$

logical structure:

$S, \alpha \models \neg\varphi$	iff $S, \alpha \not\models \varphi$
$S, \alpha \models \varphi \wedge \psi$	iff $S, \alpha \models \varphi$ and $S, \alpha \models \psi$
$S, \alpha \models \exists x.\ \varphi$	iff there exists $v \in D$ such that $S, \alpha[x \mapsto v] \models \varphi$

Example

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula

Example

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula
- Q: Is φ satisfiable?

Example

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula
- Q: Is φ satisfiable?
- A: Yes. Consider the structure S:
  - Domain: $\mathbb{N}_0$
  - Interpretation:
    - 0 and 1 are mapped to 0 and 1 in $\mathbb{N}_0$
    - $+$ means addition
    - $=$ means equality

S satisfies φ. S is said to be a model of φ.

Example (cont.)

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula

Example (cont.)

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula
- Q: Is φ valid?

Example (cont.)

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula
- Q: Is φ valid?
- A: No. Consider the structure S':
  - Domain: $\mathbb{N}_0$
  - Interpretation:
    - 0 and 1 are mapped to 0 and 1 in $\mathbb{N}_0$
    - $+$ means multiplication
    - $=$ means equality

S' does not satisfy φ.

Theories T, T-safisfiability and T-validity

- A Σ-theory T is defined by a set of Σ-sentences.

Theories T, T-safisfiability and T-validity

- A Σ-theory T is defined by a set of Σ-sentences.

- A Σ-formula φ is T-satisfiable if there exists a structure that satisfies both the sentences of T and φ.
- A Σ-formula φ is T-valid if all structures that satisfy the sentences defining T also satisfy φ.

Theories T, T-safisfiability and T-validity

- A Σ-theory T is defined by a set of Σ-sentences.

- A Σ-formula φ is T-satisfiable if there exists a structure that satisfies both the sentences of T and φ.
- A Σ-formula φ is T-valid if all structures that satisfy the sentences defining T also satisfy φ.

- The number of sentences that are necessary for defining a theory may be large or infinite.
- Instead, it is common to define a theory through a set of axioms.
- The theory is defined by these axioms and everything that can be inferred from them by a sound inference system.

Examples

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ //$=$ must be reflexive
  2. $\forall x.\ \forall y.\ x + y = y + x$ //$+$ must be commutative

Examples

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ $\quad$ $//=$ must be reflexive
  2. $\forall x.\ \forall y.\ x + y = y + x$ $\quad$ $//+$ must be commutative
- Q: Is φ T-satisfiable?

Examples

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ $\quad$ $//=$ must be reflexive
  2. $\forall x.\ \forall y.\ x + y = y + x$ $\quad$ $//+$ must be commutative
- Q: Is φ T-satisfiable?
- A: Yes, S is a model.

Examples

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ $\quad$ $//=$ must be reflexive
  2. $\forall x.\ \forall y.\ x + y = y + x$ $\quad$ $//+$ must be commutative
- Q: Is φ T-satisfiable?
- A: Yes, S is a model.
- Q: Is φ T-valid?

Examples

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ $\quad$ $//=$ must be reflexive
  2. $\forall x.\ \forall y.\ x + y = y + x$ $\quad$ $//+$ must be commutative
- Q: Is φ T-satisfiable?
- A: Yes, S is a model.
- Q: Is φ T-valid?
- A: No. S' satisfies the sentences in T but not φ.

Examples

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ $\quad$ ($=$ is reflexive)
  2. $\forall x, y, z.\ ((x = y \land y = z) \rightarrow x = z)$ $\quad$ ($=$ is transitive)
  3. $\forall x.\ \forall y.\ x + y = y + x$ $\quad$ ($+$ is commutative)
  4. $\forall x.\ 0 + x = x$ $\quad$ (0 is neutral element for $+$)

Examples

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ ($=$ is reflexive)
  2. $\forall x, y, z.\ ((x = y \land y = z) \rightarrow x = z)$ ($=$ is transitive)
  3. $\forall x.\ \forall y.\ x + y = y + x$ ($+$ is commutative)
  4. $\forall x.\ 0 + x = x$ (0 is neutral element for $+$)
- Q: Is φ T-satisfiable?

Examples

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ ($=$ is reflexive)
  2. $\forall x, y, z.\ ((x = y \land y = z) \rightarrow x = z)$ ($=$ is transitive)
  3. $\forall x.\ \forall y.\ x + y = y + x$ ($+$ is commutative)
  4. $\forall x.\ 0 + x = x$ (0 is neutral element for $+$)
- Q: Is φ T-satisfiable?
- A: Yes, S is a model.

Examples

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ ($=$ is reflexive)
  2. $\forall x, y, z.\ ((x = y \wedge y = z) \rightarrow x = z)$ ($=$ is transitive)
  3. $\forall x.\ \forall y.\ x + y = y + x$ ($+$ is commutative)
  4. $\forall x.\ 0 + x = x$ (0 is neutral element for $+$)
- Q: Is φ T-satisfiable?
- A: Yes, S is a model.
- Q: Is φ T-valid?

Examples

- $\Sigma = \{0, 1, +, =\}$
- $\varphi = \exists x.\ x + 0 = 1$ a Σ-formula.
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ ($=$ is reflexive)
  2. $\forall x, y, z.\ ((x = y \wedge y = z) \rightarrow x = z)$ ($=$ is transitive)
  3. $\forall x.\ \forall y.\ x + y = y + x$ ($+$ is commutative)
  4. $\forall x.\ 0 + x = x$ (0 is neutral element for $+$)
- Q: Is φ T-satisfiable?
- A: Yes, S is a model.
- Q: Is φ T-valid?
- A: Yes. (S' does not satisfy the fourth axiom. 3, 4 $\rightarrow \varphi$.)

Example

- $\Sigma = \{=\}$
- $\varphi = (x = y \land y \neq z) \rightarrow x \neq z$ a Σ-formula
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ (reflexivity)
  2. $\forall x.\ \forall y.\ x = y \rightarrow y = x$ (symmetry)
  3. $\forall x.\ \forall y.\ \forall z.\ x = y \land y = z \rightarrow x = z$ (transitivity)

Example

- $\Sigma = \{=\}$
- $\varphi = (x = y \land y \neq z) \rightarrow x \neq z$ a Σ-formula
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ (reflexivity)
  2. $\forall x.\ \forall y.\ x = y \rightarrow y = x$ (symmetry)
  3. $\forall x.\ \forall y.\ \forall z.\ x = y \land y = z \rightarrow x = z$ (transitivity)
- Q: Is φ T-satisfiable?

Example

- $\Sigma = \{=\}$
- $\varphi = (x = y \wedge y \neq z) \rightarrow x \neq z$ a Σ-formula
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ (reflexivity)
  2. $\forall x.\ \forall y.\ x = y \rightarrow y = x$ (symmetry)
  3. $\forall x.\ \forall y.\ \forall z.\ x = y \wedge y = z \rightarrow x = z$ (transitivity)
- Q: Is φ T-satisfiable?
- A: Yes.

Example

- $\Sigma = \{=\}$
- $\varphi = (x = y \land y \neq z) \rightarrow x \neq z$ a Σ-formula
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ (reflexivity)
  2. $\forall x.\ \forall y.\ x = y \rightarrow y = x$ (symmetry)
  3. $\forall x.\ \forall y.\ \forall z.\ x = y \land y = z \rightarrow x = z$ (transitivity)
- Q: Is φ T-satisfiable?
- A: Yes.
- Q: Is φ T-valid?

Example

- $\Sigma = \{=\}$
- $\varphi = (x = y \wedge y \neq z) \rightarrow x \neq z$ a Σ-formula
- We now define the Σ-theory T by the following axioms:
  1. $\forall x.\ x = x$ (reflexivity)
  2. $\forall x.\ \forall y.\ x = y \rightarrow y = x$ (symmetry)
  3. $\forall x.\ \forall y.\ \forall z.\ x = y \wedge y = z \rightarrow x = z$ (transitivity)
- Q: Is φ T-satisfiable?
- A: Yes.
- Q: Is φ T-valid?
- A: Yes. Every structure that satisfies T also satisfies φ.

Example

- $\Sigma = \{<\}$
- $\varphi : \forall x.\ \exists y.\ y < x$ a Σ-formula
- Consider the Σ-theory T defined by the axioms:
  1. $\forall x.\ \forall y.\ \forall z.\ x < y \land y < z \rightarrow x < z$ (transitivity)
  2. $\forall x.\ \forall y.\ x < y \rightarrow \neg(y < x)$ (anti-symmetry)

Example

- $\Sigma = \{<\}$
- $\varphi : \forall x.\ \exists y.\ y < x$ a Σ-formula
- Consider the Σ-theory T defined by the axioms:
  1. $\forall x.\ \forall y.\ \forall z.\ x < y \land y < z \rightarrow x < z$ (transitivity)
  2. $\forall x.\ \forall y.\ x < y \rightarrow \neg(y < x)$ (anti-symmetry)
- Q: Is φ T-satisfiable?

Example

- $\Sigma = \{<\}$
- $\varphi : \forall x.\ \exists y.\ y < x$ a Σ-formula
- Consider the Σ-theory T defined by the axioms:
  1. $\forall x.\ \forall y.\ \forall z.\ x < y \land y < z \rightarrow x < z$ (transitivity)
  2. $\forall x.\ \forall y.\ x < y \rightarrow \neg(y < x)$ (anti-symmetry)
- Q: Is φ T-satisfiable?
- A: Yes. We construct a model for it:
  - Domain: $\mathbb{Z}$
  - $<$ means "less than"

Example

- $\Sigma = \{<\}$
- $\varphi : \forall x.\ \exists y.\ y < x$ a Σ-formula
- Consider the Σ-theory T defined by the axioms:
  1. $\forall x.\ \forall y.\ \forall z.\ x < y \wedge y < z \rightarrow x < z$ (transitivity)
  2. $\forall x.\ \forall y.\ x < y \rightarrow \neg(y < x)$ (anti-symmetry)
- Q: Is φ T-satisfiable?
- A: Yes. We construct a model for it:
  - Domain: $\mathbb{Z}$
  - $<$ means "less than"
- Q: Is φ T-valid?

Example

- $\Sigma = \{<\}$
- $\varphi : \forall x.\ \exists y.\ y < x$ a Σ-formula
- Consider the Σ-theory T defined by the axioms:
  1. $\forall x.\ \forall y.\ \forall z.\ x < y \wedge y < z \rightarrow x < z$ (transitivity)
  2. $\forall x.\ \forall y.\ x < y \rightarrow \neg(y < x)$ (anti-symmetry)
- Q: Is φ T-satisfiable?
- A: Yes. We construct a model for it:
  - Domain: $\mathbb{Z}$
  - $<$ means "less than"
- Q: Is φ T-valid?
- A: No. We construct a structure to the contrary:
  - Domain: $\mathbb{N}_0$
  - $<$ means "less than"

Logic fragments

- So far we only restricted the non-logical symbols by signatures and their interpretation by theories.
- Sometimes we want to restrict the grammar and the logical symbols that we can use as well.
- These are called logic fragments.
- Examples:
  - The quantifier-free fragment over $\Sigma = \{0, 1, +, =\}$
  - The conjunctive fragment over $\Sigma = \{0, 1, +, =\}$

Fragments

Fragments

- Q: Which FO theory is propositional logic?

Fragments

- Q: Which FO theory is propositional logic?
- A: The quantifier-free fragment of the FO theory with signature $\Sigma = \{x_1, x_2, \ldots, \mathit{identity}\}$ with variables $x_1, x_2, \ldots$, the unary *identity* predicate (which we skip in the syntax), and without axioms.

Fragments

- Q: Which FO theory is propositional logic?
- A: The quantifier-free fragment of the FO theory with signature $\Sigma = \{x_1, x_2, \ldots, \mathit{identity}\}$ with variables $x_1, x_2, \ldots$, the unary *identity* predicate (which we skip in the syntax), and without axioms.

  Example: $x_1 \rightarrow (x_2 \vee x_3)$

  Thus, propositional logic is also a first-order theory.
  (A very degenerate one.)

Fragments

- Q: Which FO theory is propositional logic?
- A: The quantifier-free fragment of the FO theory with signature $\Sigma = \{x_1, x_2, \ldots, \mathit{identity}\}$ with variables $x_1, x_2, \ldots$, the unary *identity* predicate (which we skip in the syntax), and without axioms.

  Example: $x_1 \rightarrow (x_2 \vee x_3)$

  Thus, propositional logic is also a first-order theory.
  (A very degenerate one.)

- Q: What if we allow quantifiers?

Fragments

- Q: Which FO theory is propositional logic?
- A: The quantifier-free fragment of the FO theory with signature $\Sigma = \{x_1, x_2, \ldots, \mathit{identity}\}$ with variables $x_1, x_2, \ldots$, the unary *identity* predicate (which we skip in the syntax), and without axioms.

  Example: $x_1 \to (x_2 \vee x_3)$

  Thus, propositional logic is also a first-order theory.
  (A very degenerate one.)

- Q: What if we allow quantifiers?
- A: We get the theory of quantified boolean formulas (QBF).
  Example:
  - $\forall x_1.\ \exists x_2.\ \forall x_3.\ x_1 \to (x_2 \vee x_3)$

Some famous theories

- Presburger arithmetic: $\Sigma = \{0, 1, +, >\}$ over integers
- Peano arithmetic: $\Sigma = \{0, 1, +, *, >\}$ over integers
- Linear real arithmetic: $\Sigma = \{0, 1, +, >\}$ over reals
- Real arithmetic: $\Sigma = \{0, 1, +, *, >\}$ over reals
- Theory of arrays
- Theory of pointers
- ...

The algorithmic point of view...

- It is also common to present theory fragments via an abstract grammar rather than restrictions on the generic first-order grammar.
- We assume that the interpretation of symbols is fixed to their common use.
  - Thus $+$ is plus, ...

The algorithmic point of view...

- Example: Equality logic
- Grammar:

$$\begin{aligned} \mathit{formula} &::= \mathit{atom} \mid \mathit{formula} \wedge \mathit{formula} \mid \neg \mathit{formula} \\ \mathit{atom} &::= \mathit{Boolean\text{-}variable} \mid \\ &\quad\ \mathit{variable} = \mathit{variable} \mid \\ &\quad\ \mathit{variable} = \mathit{constant} \mid \\ &\quad\ \mathit{constant} = \mathit{constant} \end{aligned}$$

- Interpretation: $=$ is equality.

Expressivity of a theory

- Each formula defines a language:
  The set of satisfying assignments (models) are the words accepted by this language.

- Consider the fragment '2-CNF':

$$\begin{array}{lcl} \textit{formula} & ::= & (\textit{literal} \vee \textit{literal}) \mid \textit{formula} \wedge \textit{formula} \\ \textit{literal} & ::= & \textit{Boolean-variable} \mid \neg \textit{Boolean-variable} \end{array}$$

- Example formula:

$$(x_1 \vee \neg x_2) \wedge (\neg x_3 \vee x_2)$$

Expressivity of a theory

- Now consider the propositional logic formula $\varphi = (x_1 \vee x_2 \vee x_3)$
- Q: Can we express this language with 2-CNF?

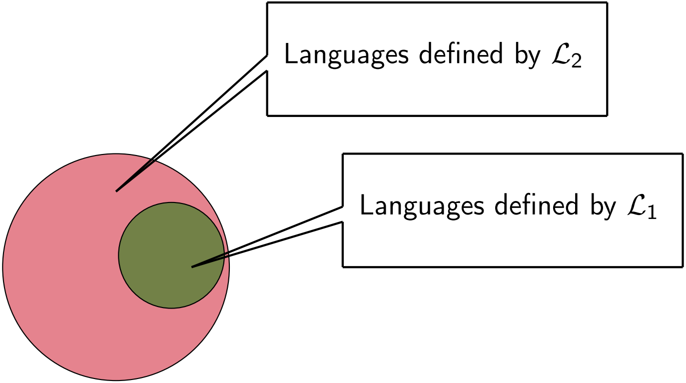
Expressivity of a theory

- Now consider the propositional logic formula $\varphi = (x_1 \vee x_2 \vee x_3)$
- Q: Can we express this language with 2-CNF?
- A: No.
- Proof:

Expressivity of a theory

- Now consider the propositional logic formula $\varphi = (x_1 \vee x_2 \vee x_3)$
- Q: Can we express this language with 2-CNF?
- A: No.
- Proof:
  - The language accepted by φ has 7 words: all assignments other than $x_1 = x_2 = x_3 = 0$ (*false*).
  - A 2-CNF clause removes 2 assignments, which leaves us with 6 accepted words.
    E.g., $(x_1 \vee x_2)$ removes the assignments $x_1 = x_2 = x_3 = 0$ and $x_1 = x_2 = 0$, $x_3 = 1$.
  - Additional clauses only remove more assignments.

Examples



$\mathcal{L}_2$ is more expressive than $\mathcal{L}_1$. Notation: $\mathcal{L}_1 \prec \mathcal{L}_2$.

- Claim: 2-CNF $\prec$ propositional logic.
- Generally there is only a partial order between theories.

The tradeoff

- So we see that theories can have different expressive power.
- Q: Why would we want to restrict ourselves to a theory or a fragment? Why not take some 'maximal theory'?

The tradeoff

- So we see that theories can have different expressive power.
- Q: Why would we want to restrict ourselves to a theory or a fragment? Why not take some 'maximal theory'?
- A: Adding axioms to the theory may make it harder to decide or even undecidable.

Example: Resolution

$$\frac{(x \vee l_1 \vee \ldots \vee l_n) \qquad (\neg x \vee l'_1 \vee \ldots \vee l'_m)}{(l_1 \vee \ldots \vee l_n \vee l'_1 \vee \ldots \vee l'_m)} \;(\mathit{Resolution})$$

- Resolution is a sound and complete proof system for CNF-formulas (of propositional logic).
- This means that with resolution we can prove any valid propositional CNF formula, and only such formulas. The proof is finite.

Example: Resolution

$$\frac{(x \vee l_1 \vee \ldots \vee l_n) \qquad (\neg x \vee l'_1 \vee \ldots \vee l'_m)}{(l_1 \vee \ldots \vee l_n \vee l'_1 \vee \ldots \vee l'_m)} \;(\textit{Resolution})$$

- Resolution is a sound and complete proof system for CNF-formulas (of propositional logic).
- This means that with resolution we can prove any valid propositional CNF formula, and only such formulas. The proof is finite.
- But there are first-order theories for which there exists no sound and complete proof system.

Example: First-order Peano arithmetic

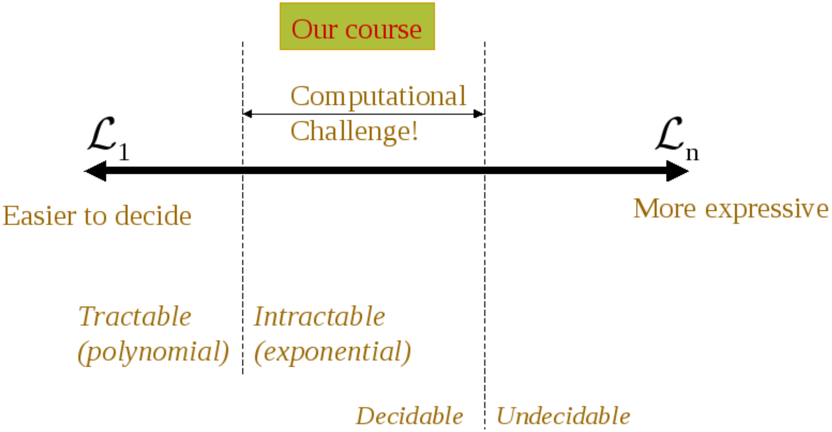
- $\Sigma = \{0, 1, +, *, =\}$
- Domain: Natural numbers
- Axioms ("semantics"):
  1. $\forall x.\ (x \neq x + 1)$
  2. $\forall x.\ \forall y.\ (x \neq y) \rightarrow (x + 1 \neq y + 1)$
  3. Induction
  4. $\forall x.\ x + 0 = x$
  5. $\forall x.\ \forall y : (x + y) + 1 = x + (y + 1)$
  6. $\forall x.\ x * 0 = 0$
  7. $\forall x.\ \forall y.\ x * (y + 1) = x * y + x$

UNDECIDABLE!

- $\Sigma = \{0, 1, +, \not{*}, =\}$
- Domain: Natural numbers
- Axioms ("semantics"):
  1. $\forall x.\ (x \neq x + 1)$
  2. $\forall x.\ \forall y.\ (x \neq y) \rightarrow (x + 1 \neq y + 1)$
  3. Induction
  4. $\forall x.\ x + 0 = x$
  5. $\forall x.\ \forall y.\ (x + y) + 1 = x + (y + 1)$
  6. ~~$\forall x.\ x * 0 = 0$~~
  7. ~~$\forall x.\ \forall y.\ x * (y + 1) = x * y + x$~~

DECIDABLE!

Tradeoff: Expressivity vs. computational hardness



In this lecture we assume $P \neq NP$.

When is a specific theory useful?

- Expressible enough to state something interesting.
- Decidable (or semi-decidable) and more efficiently solvable than richer theories.
- More expressible, or more natural for expressing some models in comparison to 'leaner' theories.

Expressivity and complexity

- Q1: Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two theories whose satisfiability problem is decidable and in the same complexity class. Is the satisfiability problem of an $\mathcal{L}_1$ formula reducible to a satisfiability problem of an $\mathcal{L}_2$ formula?

Expressivity and complexity

- Q1: Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two theories whose satisfiability problem is decidable and in the same complexity class. Is the satisfiability problem of an $\mathcal{L}_1$ formula reducible to a satisfiability problem of an $\mathcal{L}_2$ formula?

  A1: Yes, reduction with the given complexity is possible.

Expressivity and complexity

- Q1: Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two theories whose satisfiability problem is decidable and in the same complexity class. Is the satisfiability problem of an $\mathcal{L}_1$ formula reducible to a satisfiability problem of an $\mathcal{L}_2$ formula?

  A1: Yes, reduction with the given complexity is possible.
- Q2: Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two theories whose satisfiability problems are reducible to each other. Are $\mathcal{L}_1$ and $\mathcal{L}_2$ in the same complexity class?

Expressivity and complexity

- Q1: Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two theories whose satisfiability problem is decidable and in the same complexity class. Is the satisfiability problem of an $\mathcal{L}_1$ formula reducible to a satisfiability problem of an $\mathcal{L}_2$ formula?

  A1: Yes, reduction with the given complexity is possible.

- Q2: Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two theories whose satisfiability problems are reducible to each other. Are $\mathcal{L}_1$ and $\mathcal{L}_2$ in the same complexity class?

  A2: It depends on the complexity of the reduction.