Satisfiability Checking

First Order Logic

Prof. Dr. Erika Ábrahám

Theory of Hybrid Systems
Informatik 2

WS 11/12

From informal to formal logics

- We have seen that natural languages are not well-suited for correct reasoning.

From informal to formal logics

- We have seen that natural languages are not well-suited for correct reasoning.
- Assume the argumentation:
  1. All women love shopping.
  2. Eve is a woman.
  3. Thus Eve loves shopping.

From informal to formal logics

- We have seen that natural languages are not well-suited for correct reasoning.
- Assume the argumentation:
  1. All women love shopping.
  2. Eve is a woman.
  3. Thus Eve loves shopping.
- We can formalize it by defining

From informal to formal logics

- We have seen that natural languages are not well-suited for correct reasoning.
- Assume the argumentation:
  1. All women love shopping.
  2. Eve is a woman.
  3. Thus Eve loves shopping.
- We can formalize it by defining

  constants: Eve

  variables: x

  predicate symbols: $isWoman(\cdot)$, $lovesShopping(\cdot)$

From informal to formal logics

- We have seen that natural languages are not well-suited for correct reasoning.
- Assume the argumentation:
  1. All women love shopping.
  2. Eve is a woman.
  3. Thus Eve loves shopping.
- We can formalize it by defining

  constants: Eve

  variables: x

  predicate symbols: $isWoman(\cdot)$, $lovesShopping(\cdot)$

  Assume

From informal to formal logics

- We have seen that natural languages are not well-suited for correct reasoning.
- Assume the argumentation:
  1. All women love shopping.
  2. Eve is a woman.
  3. Thus Eve loves shopping.
- We can formalize it by defining

  constants: Eve

  variables: x

  predicate symbols: $isWoman(\cdot)$, $lovesShopping(\cdot)$

  Assume

  1. $\forall x.\ isWoman(x) \rightarrow lovesShopping(x)$

From informal to formal logics

- We have seen that natural languages are not well-suited for correct reasoning.
- Assume the argumentation:
  1. All women love shopping.
  2. Eve is a woman.
  3. Thus Eve loves shopping.
- We can formalize it by defining

  constants: Eve
  variables: x
  predicate symbols: $isWoman(\cdot)$, $lovesShopping(\cdot)$

  Assume
  1. $\forall x.\ isWoman(x) \rightarrow lovesShopping(x)$
  2. $isWoman(Eve)$

From informal to formal logics

- We have seen that natural languages are not well-suited for correct reasoning.
- Assume the argumentation:
  1. All women love shopping.
  2. Eve is a woman.
  3. Thus Eve loves shopping.
- We can formalize it by defining

  constants: Eve
  variables: x
  predicate symbols: $isWoman(\cdot)$, $lovesShopping(\cdot)$

  Assume
  1. $\forall x.\ isWoman(x) \rightarrow lovesShopping(x)$
  2. $isWoman(Eve)$

  Then

From informal to formal logics

- We have seen that natural languages are not well-suited for correct reasoning.
- Assume the argumentation:
  1. All women love shopping.
  2. Eve is a woman.
  3. Thus Eve loves shopping.
- We can formalize it by defining

  constants: Eve
  variables: x
  predicate symbols: $isWoman(\cdot)$, $lovesShopping(\cdot)$

  Assume
  1. $\forall x.\ isWoman(x) \rightarrow lovesShopping(x)$
  2. $isWoman(Eve)$

  Then
  3. $lovesShopping(Eve)$

First-Order Logic

- First-order (FO) logic is a framework.
- It gives us a generic syntax by recursively defining strings on an alphabet.
- Non-logical elements are logically combined using:
  - constants (Eve, 0, true, . . .)
  - variables (x,y,. . .)
  - function symbols ($+(\cdot,\cdot)$, vaterOf$(\cdot)$,. . .)
  - predicate symbols ($>(\cdot,\cdot)$, isPrime$(\cdot)$, isBrotherOf$(\cdot,\cdot)$,. . .)
  - logical symbols $((,),\wedge,\neg,\ldots,\exists,\forall)$.

First-Order Logic

- First-order (FO) logic is a framework.
- It gives us a generic syntax by recursively defining strings on an alphabet.
- Non-logical elements are logically combined using:
  - constants (Eve, 0, true, ...)
  - variables (x,y,...)
  - function symbols ($+(\cdot,\cdot)$, vaterOf$(\cdot)$,...)
  - predicate symbols ($>(\cdot,\cdot)$, isPrime$(\cdot)$, isBrotherOf$(\cdot,\cdot)$,...)
  - logical symbols $((,), \wedge, \neg, \ldots, \exists, \forall)$.

Note:

- Constants can also be seen as function symbols of arity 0.
- Sometimes equality $(=)$ is included as a logical symbol.
- E.g., the Boolean 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.

Formation rules

Formation rules

Terms are inductively defined by the following rules:

Formation rules

Terms are inductively defined by the following rules:

1. All constants and variables are terms.

Formation rules

Terms 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.

Formation rules

Terms 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.

Formation rules

Terms 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.

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

Formation rules

Terms 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.

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

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 formula.

Formation rules

Terms 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.

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

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 formula.
2. If φ is a formula, then $(\neg\varphi)$ is a formula.

Formation rules

Terms 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.

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

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 formula.
2. If φ is a formula, then $(\neg\varphi)$ is a formula.
3. If φ and ψ are formulae, then $(\varphi \wedge \psi)$ is a formula.

Formation rules

Terms 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.

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

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 formula.
2. If φ is a formula, then $(\neg\varphi)$ is a formula.
3. If φ and ψ are formulae, then $(\varphi \wedge \psi)$ is a formula.
4. Similar rules apply to other binary logical connectives.

Formation rules

Terms 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.

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

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 formula.
2. If φ is a formula, then $(\neg\varphi)$ is a formula.
3. If φ and ψ are formulae, 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 formulae.

Formation rules

Terms 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.

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

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 formula.
2. If φ is a formula, then $(\neg\varphi)$ is a formula.
3. If φ and ψ are formulae, 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 formulae.

Only expressions which can be obtained by finitely many applications of rules 1–5 are formulae.

Formation rules

Terms 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.

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

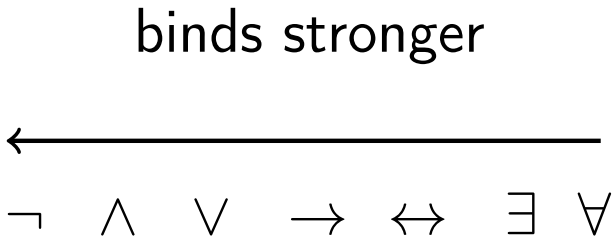
1. If P is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms then $P(t_1, \ldots, t_n)$ is a formula.
2. If φ is a formula, then $(\neg\varphi)$ is a formula.
3. If φ and ψ are formulae, then $(\varphi \wedge \psi)$ is a formula.
4. Similar rules apply to other binary logical connectives.
5. If φ is a formula and x is a variable, then $(\forall x.\varphi)$ and $(\exists x.\ \varphi)$ are formulae.

Only expressions which can be obtained by finitely many applications of rules 1–5 are formulae.

The formulae obtained by the first rule are said to be atomic.

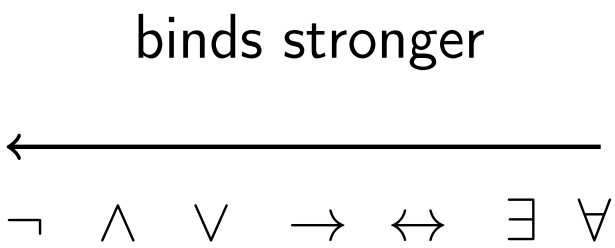
Notational conventions

We omit parenthesis whenever we may restore them through operator precedence:



Notational conventions

We omit parenthesis whenever we may restore them through operator precedence:



- Thus, we write:

  $\neg\neg a$ for $(\neg(\neg a))$,

  $\exists a.\ \exists b.\ (a \wedge b \rightarrow F(a,b))$ for $\exists a.\ \exists b.\ ((a \wedge b) \rightarrow F(a,b))$

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:

Free and bound variables

Examples:

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

Freeness and boundness can be also specialized to specific occurrences of variables in a formula. In $z \vee \forall z.P(z)$, the first occurrence of z is free while the second is bound.

Some definitions

- A signature Σ fixes a set of non-logical symbols.
- A Σ-formula is a formula with non-logical symbols from Σ.
- A variable is free if it is not bound by a quantifier.
- A sentence is a formula without free variables.

Some definitions

- A signature Σ fixes a set of non-logical symbols.
- A Σ-formula is a formula with non-logical symbols from Σ.
- A variable is free if it is not bound by a quantifier.
- A sentence is a formula without free variables.

In the previous example:

Some definitions

- A signature Σ fixes a set of non-logical symbols.
- A Σ-formula is a formula with non-logical symbols from Σ.
- A variable is free if it is not bound by a quantifier.
- A sentence is a formula without free variables.

In the previous example:
$\Sigma = (\mathit{Eve}, \mathit{isWoman}(\cdot), \mathit{lovesShopping}(\cdot))$ with

- *Eve* a constant and
- *isWoman* and *lovesShopping* unary predicate symbols.

The formulae

1. $\forall x.\ \mathit{isWoman}(x) \rightarrow \mathit{lovesShopping}(x)$
2. *isWoman(Eve)*
3. *lovesShopping(Eve)*

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

Examples

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

Examples

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

Examples

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

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

Examples

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

Examples

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

$$\forall n.\ \exists p.\ 1 < n \rightarrow \mathit{isPrime}(p) \wedge n < p < 2 * n$$

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 structure is given by:
  - a domain D,
  - an interpretation I of the non-logical symbols that
    - maps each constant symbol to a domain element,
    - maps each function symbol to a function of the same arity, and
    - maps each predicate symbol to a predicate of the same arity,
  - an assignment of a domain element to each free (unquantified) variable.

Structures, satisfiability, validity

- A structure is given by:
  - a domain D,
  - an interpretation I of the non-logical symbols that
    - maps each constant symbol to a domain element,
    - maps each function symbol to a function of the same arity, and
    - maps each predicate symbol to a predicate of the same arity,
  - an assignment of a domain element to each free (unquantified) variable.
- A formula is satisfiable if there exists a structure that satisfies it.

Structures, satisfiability, validity

- A structure is given by:
  - a domain D,
  - an interpretation I of the non-logical symbols that
    - maps each constant symbol to a domain element,
    - maps each function symbol to a function of the same arity, and
    - maps each predicate symbol to a predicate of the same arity,
  - an assignment of a domain element to each free (unquantified) variable.
- A formula is satisfiable if there exists a structure that satisfies it.
- A formula is valid if it is satisfied by all structures.

Semantics

Semantics of terms and formulae under a structure (D, I):

Semantics

Semantics of terms and formulae under a structure (D, I):

constants: $[\![c]\!]_{(D,I)} = I(c)$

variables: $[\![x]\!]_{(D,I)} = I(x)$

functions: $[\![f(t_1, \ldots, t_n)]\!]_{(D,I)} = I(f)([\![t_1]\!]_{(D,I)}, \ldots, [\![t_n]\!]_{(D,I)})$

predicates: $[\![p(t_1, \ldots, t_n)]\!]_{(D,I)} = I(p)([\![t_1]\!]_{(D,I)}, \ldots, [\![t_n]\!]_{(D,I)})$

logical structure:

$$[\![\neg\varphi]\!]_{(D,I)} = \begin{cases} \mathit{true} & \mathit{if}\ [\![\varphi]\!]_{(D,I)} = \mathit{false} \\ \mathit{false} & \mathit{if}\ [\![\varphi]\!]_{(D,I)} = \mathit{true} \end{cases}$$

$$[\![\varphi \wedge \psi]\!]_{(D,I)} = \begin{cases} \mathit{true} & \mathit{if}\ [\![\varphi]\!]_{(D,I)} = \mathit{true}\ \mathit{and}\ [\![\psi]\!]_{(D,I)} = \mathit{true} \\ \mathit{false} & \mathit{if}\ [\![\varphi]\!]_{(D,I)} = \mathit{false}\ \mathit{or}\ [\![\psi]\!]_{(D,I)} = \mathit{false} \end{cases}$$

$$[\![\exists x.\ \varphi]\!]_{(D,I)} = \begin{cases} \mathit{true} & \mathit{if\ there\ exists}\ v \in D\ \mathit{such\ that}\ [\![\varphi]\!]_{(D,I[x \mapsto v])} = \mathit{true} \\ \mathit{false} & \mathit{if\ for\ all}\ v \in D\ \mathit{we\ have}\ [\![\varphi]\!]_{(D,I[x \mapsto v])} = \mathit{false} \end{cases}$$

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 equality
    - $+$ means addition

  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 equality
    - $+$ means multiplication

  S' does not satisfy φ.

Theories, T-safisfiability and T-validity

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

Theories, T-safisfiability and T-validity

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

- 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.

Theories, T-safisfiability and T-validity

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

- 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.

- 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.

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

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
- 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. $\forall x.\ x = x$ // $=$ must be reflexive
  2. $\forall x.\ \forall y.\ x + y = y + x$ //$+$ 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. $\forall x.\ x = x$ // $=$ must be reflexive
  2. $\forall x.\ \forall y.\ x + y = y + x$ //$+$ 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. $\forall x.\ x = x \quad //=$ must be reflexive
  2. $\forall x.\ \forall y.\ x + y = y + x \quad //+$ must be commutative
  3. $\forall x.\ 0 + x = x$

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
  3. $\forall x.\ 0 + x = x$
- 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. $\forall x.\ x = x$ // $=$ must be reflexive
  2. $\forall x.\ \forall y.\ x + y = y + x$ //$+$ must be commutative
  3. $\forall x.\ 0 + x = x$
- 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
  3. $\forall x.\ 0 + x = x$
- 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
  3. $\forall x.\ 0 + x = x$
- Q: Is φ T-satisfiable?
- A: Yes, S is a model.
- Q: Is φ T-valid?
- A: Yes. (S' does not satisfy the third axiom.)

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)

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?

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

- $\Sigma = \{=\}$
- $\varphi = (x = y \wedge y \neq z) \rightarrow x \neq z$ 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?

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. $\forall x.\ \forall y.\ \forall z.\ x < y \wedge y < z \rightarrow x < z$ (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. $\forall x.\ \forall y.\ \forall z.\ x < y \wedge y < z \rightarrow x < z$ (transitivity)
  2. $\forall x.\ \forall y.\ x < y \rightarrow \neg(y < x)$ (anti-symmetry)
- Q: Is φ T-satisfiable?

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"

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

- Let T be a theory with $\Sigma = \{\}$ without axioms.

Fragments

- Let T be a theory with $\Sigma = \{\}$ without axioms.
- Q: What is the quantifier-free frament of T?

Fragments

- Let T be a theory with $\Sigma = \{\}$ without axioms.
- Q: What is the quantifier-free frament of T?
- A: Propositional logic

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

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

Fragments

- Let T be a theory with $\Sigma = \{\}$ without axioms.
- Q: What is the quantifier-free frament of T?
- A: Propositional logic

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

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

- Q: What is T?

Fragments

- Let T be a theory with $\Sigma = \{\}$ without axioms.
- Q: What is the quantifier-free frament of T?
- A: Propositional logic

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

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

- Q: What is T?
- A: Quantified Boolean formulae (QBF)

  Example:
  - $\forall x_1.\ \exists x_2.\ \forall x_3.\ x_1 \rightarrow (x_2 \vee x_3)$

Some famous theories

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

The algorithmic point of view...

- It is also common to present theories NOT through the axioms that define them.
- The interpretation of symbols is fixed to their common use.
  - Thus $+$ is plus, ...
- The fragment is defined via grammar rules rather than restrictions on the generic first-order grammar.

The algorithmic point of view...

- Example: Equality logic
- Grammar:

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

- Interpretation: $=$ is equality.

Expressiveness 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:

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

Expressiveness 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?

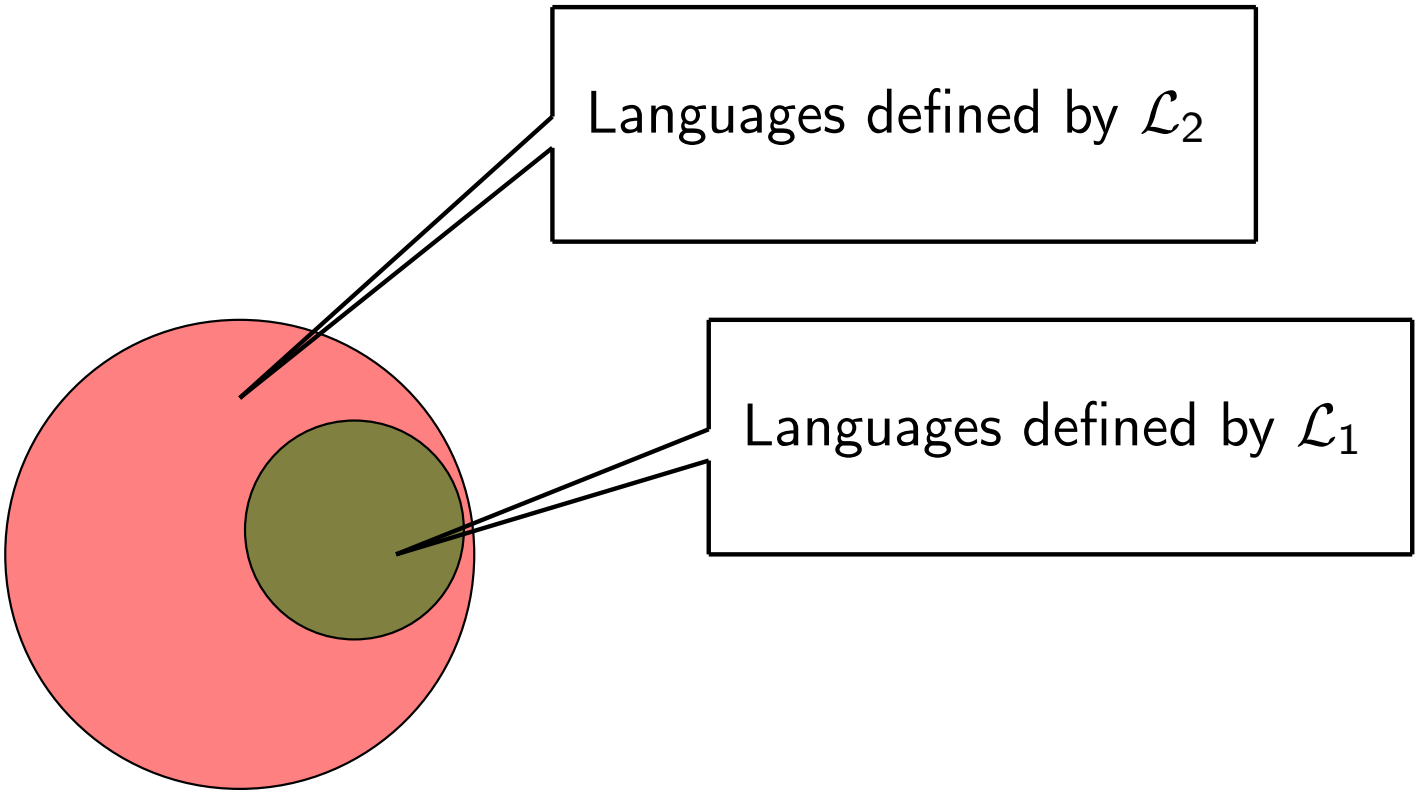
Expressiveness of a theory

- 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:

Expressiveness 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: Hilbert axiom system ($\mathcal{H}$)

- Let $\mathcal{H}$ be (M.P) + the following axiom schemas:

$$\frac{}{A \rightarrow (B \rightarrow A)} \ (H1)$$

$$\frac{}{((A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))} \ (H2)$$

$$\frac{}{(\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)} \ (H3)$$

- $\mathcal{H}$ is sound and complete for propositional logic.
- This means that with $\mathcal{H}$ we can prove any valid propositional formula, and only such formulae. The proof is finite.

Example

- But there exist first order theories defined by axioms which are not sufficient for proving all T-valid formulae.

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. $\forall x.\ \forall y.\ x * (y + 1) = x * y + x$

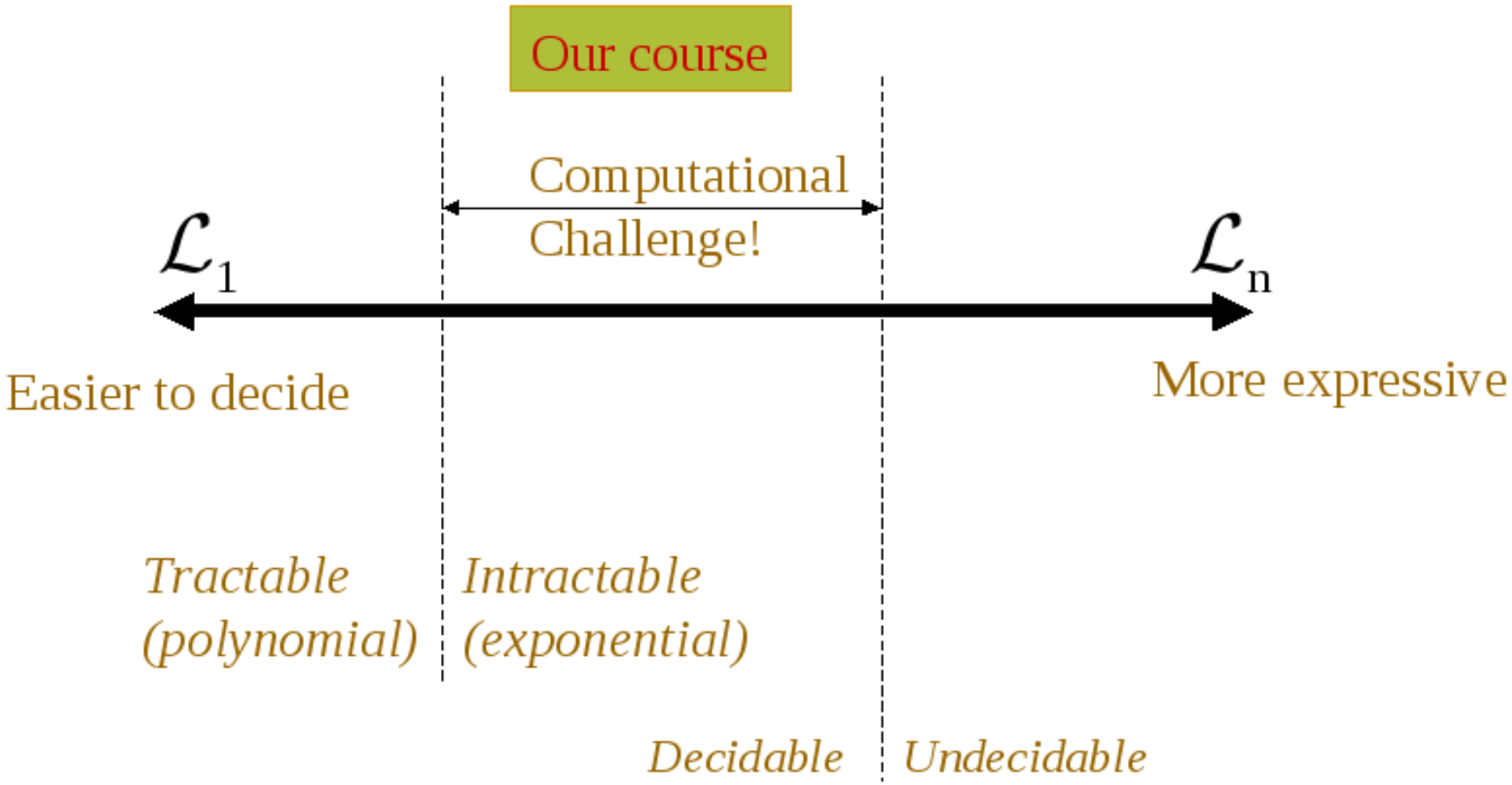
UNDECIDABLE!

Reduction: Peano Arithmetic to Presburger Arithmetic

- $\Sigma = \{0, 1, +, \cancel{*}, =\}$
- Domain: Natural numbers
- Axioms ("semantics"):
  1. $\forall x.\ (\neq x + 1)$
  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: expressiveness/computational hardness



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.

Expressiveness 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?

Expressiveness 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?
- 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?