Logic in Computer Science (ECS666U/7018P/U)

Nikos Tzevelekos

Lecture 9
Predicate Logic II

Recap: Syntax of Predicate Logic

- Vocabulary V is a set of function and relation symbols
- Terms t and formulas φ

$$t \;=\; x \mid f(t, \ldots, t)$$

$$\varphi \;=\; P(t, \ldots, t) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \to \varphi \mid \forall x.\varphi \mid \exists x.\varphi$$

where

- x ranges over variables
- f ranges over function symbols in V (nullary functions are also called constants)
- P ranges over relation symbols in V

Notes on quantification:

- "for all x satisfying $P(x)$, φ holds" translates to: $\forall x.P(x) \to \varphi$
- "there is x satisfying $P(x)$ such that φ holds" to: $\exists x.P(x) \wedge \varphi$
- "$\forall x, y.$" stands for "$\forall x.\forall y.$", and "$\exists x, y.$" stands for "$\exists x.\exists y.$"

Notations and Terminology for Syntax

Free and bound variables:

- Occurrences of variables in formulas are free; unless in the scope of a $\forall$ or $\exists$ quantifier, in which case they are bound
- We can rename all occurrences of a bound variable without changing the meaning of the formula
- A closed formula (i.e. without free variables) is also called a sentence

Notations and Terminology for Syntax

Free and bound variables:

- Occurrences of variables in formulas are free; unless in the scope of a $\forall$ or $\exists$ quantifier, in which case they are bound
- We can rename all occurrences of a bound variable without changing the meaning of the formula
- A closed formula (i.e. without free variables) is also called a sentence

Examples ($V = \{c, P, Q\}$ where c is a constant, P and Q are relations):

- $P(x) \to \forall y.Q(y)$ variable x is free and y is bound
- $P(x) \to \forall y.Q(y)$ is a formula not a sentence
- $\exists x.P(x) \to \forall y.Q(y)$ is a sentence
- $P(c) \to \forall y.Q(y)$ is a sentence because c is not a variable

Recap: Models

To give a semantics to a predicate logic formula, we need a model M:

1. a universe U, i.e. a set where terms take values
2. an interpretation of every symbol in V to its meaning:
   - the meaning of a function symbol f of arity k is a function $f_M \colon U^k \to U$
   - the meaning of a relation symbol P of arity k is a relation $P_M \subseteq U^k$
     - so, $P_M(u_1, \ldots, u_k)$ holds just if the k-tuple $(u_1, \ldots, u_k)$ is in P_M
3. an Evaluation E, i.e. a map from variables to U:
   - $E(x)$ denotes the individual in U that x is mapped to by E
   - $E[x \mapsto u]$ denotes a new evaluation that maps
     - variable x to individual u, and
     - any other variable y in the same way as E

Recap: Semantics

Given model M and evaluation E, we define $M, E \models \varphi$ (read "M, E satisfy φ"):

- if $\varphi = P(t_1, \ldots, t_k)$ then $M, E \models \varphi$ holds if $(u_1, \ldots, u_k) \in P_M$ where each u_i is the interpretation of t_i using M, E

Recap: Semantics

Given model M and evaluation E, we define $M, E \models \varphi$ (read "M, E satisfy φ"):

- if $\varphi = P(t_1, \ldots, t_k)$ then $M, E \models \varphi$ holds if $(u_1, \ldots, u_k) \in P_M$ where each u_i is the interpretation of t_i using M, E
- if φ is a boolean combination of formulas:
  - $M, E \models \neg\varphi$ holds if $M, E \models \varphi$ does not hold
  - $M, E \models \varphi_1 \wedge \varphi_2$ holds if $M, E \models \varphi_1$ and $M, E \models \varphi_2$ hold
  - $M, E \models \varphi_1 \vee \varphi_2$ holds if $M, E \models \varphi_1$ or $M, E \models \varphi_2$ holds
  - $M, E \models \varphi_1 \rightarrow \varphi_2$ holds if $M, E \models \varphi_1$ doesn't hold or $M, E \models \varphi_2$ holds

Recap: Semantics

Given model M and evaluation E, we define $M, E \models \varphi$ (read "M, E satisfy φ"):

- if $\varphi = P(t_1, \ldots, t_k)$ then $M, E \models \varphi$ holds if $(u_1, \ldots, u_k) \in P_M$ where each u_i is the interpretation of t_i using M, E
- if φ is a boolean combination of formulas:
  - $M, E \models \neg\varphi$ holds if $M, E \models \varphi$ does not hold
  - $M, E \models \varphi_1 \wedge \varphi_2$ holds if $M, E \models \varphi_1$ and $M, E \models \varphi_2$ hold
  - $M, E \models \varphi_1 \vee \varphi_2$ holds if $M, E \models \varphi_1$ or $M, E \models \varphi_2$ holds
  - $M, E \models \varphi_1 \rightarrow \varphi_2$ holds if $M, E \models \varphi_1$ doesn't hold or $M, E \models \varphi_2$ holds
- if φ starts with a quantifier:
  - $M, E \models \forall x.\varphi_1$ holds if, for all $u \in U$: $M, E[x \mapsto u] \models \varphi_1$ holds
  - $M, E \models \exists x.\varphi_1$ holds if, for some $u \in U$: $M, E[x \mapsto u] \models \varphi_1$ holds

Example (Binary Search Trees)

Vocabulary $V = \{root, key, R, B, N, <, =, Left, Right\}$ where:

- *root* is a constant symbol (i.e. nullary function)
- *key* is a unary function
- R, B and N are unary relation symbols
- $<$, $=$, *Left* and *Right* are binary relation symbols

The intended meaning of these symbols is:

- *root* is the root node of a binary search tree
- $key(x)$ is the key associated with x (e.g. a number)
- $R(x)$: x is marked Red
- $B(x)$: x is marked Black
- $N(x)$: x is a node of a binary search tree
- $x < y$: x is less than y
- $Left(x, y)$: the left child of x is y
- $Right(x, y)$: the right child of x is y

Example sentences

Encode the following sentences in predicate logic:

1. If a node is not red, then it is black.

2. Each node's key is greater than that of its left child.

3. If a node is the root node then it has a left child.

4. The parent of each black node is red.

5. There is exactly one black node.

Example sentences

Encode the following sentences in predicate logic:

1. If a node is not red, then it is black.

   $\forall x.(N(x) \wedge \neg R(x)) \rightarrow B(x)$

2. Each node's key is greater than that of its left child.

   $\forall x, y.Left(x, y) \rightarrow key(y) < key(x)$

3. If a node is the root node then it has a left child.

   $\forall x.x = root \rightarrow \exists y.Left(x, y)$

4. The parent of each black node is red.

   $\forall x, y.B(x) \wedge (Left(y, x) \vee Right(y, x)) \rightarrow R(y)$

5. There is exactly one black node.

   $\exists x.B(x) \wedge \forall y.B(y) \rightarrow x = y$

Semantics

We now consider the model M with:

- Universe $U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7\} \cup \{0, 1, 2, \ldots\}$
- Interpretation of constant symbol is $root_M = u_1$.
- Interpretation of relation symbols:

$$N_M = \{u_1, \ldots, u_7\} \qquad R_M = \{u_2, u_3\}$$
$$B_M = \{u_1, u_4, u_5, u_6, u_7\} \qquad Left_M = \{(u_1, u_2), (u_2, u_4), (u_3, u_6)\}$$
$$Right_M = \{(u_1, u_3), (u_2, u_5), (u_3, u_7)\} \qquad <_M \text{ is the less-than relation}$$

- Interpretation of function symbol *key* is:

$$key_M(u_1) = 4 \quad key_M(u_2) = 2 \quad key_M(u_3) = 6 \quad key_M(u_4) = 1$$
$$key_M(u_5) = 3 \quad key_M(u_6) = 5 \quad key_M(u_7) = 7 \quad key_M(u) = 0 \text{ for other u}$$

Question: which of the previous formulas are satisfied by M?

De Morgan Laws — Duality Principle

Duality between logic operators

- De Morgan Law: for any predicate logic formulas φ_1 and φ_2

  $\neg(\varphi_1 \wedge \varphi_2)$ is equivalent to $\neg\varphi_1 \vee \neg\varphi_2$

  $\neg(\varphi_1 \vee \varphi_2)$ is equivalent to $\neg\varphi_1 \wedge \neg\varphi_2$

  $\neg\exists x.\varphi$ is equivalent to $\forall x.\neg\varphi$

  $\neg\forall x.\varphi$ is equivalent to $\exists x.\neg\varphi$

Duality between logic operators

- De Morgan Law: for any predicate logic formulas φ_1 and φ_2

  $\neg(\varphi_1 \wedge \varphi_2)$ is equivalent to $\neg\varphi_1 \vee \neg\varphi_2$

  $\neg(\varphi_1 \vee \varphi_2)$ is equivalent to $\neg\varphi_1 \wedge \neg\varphi_2$

  $\neg\exists x.\varphi$ is equivalent to $\forall x.\neg\varphi$

  $\neg\forall x.\varphi$ is equivalent to $\exists x.\neg\varphi$

- Example: $\neg\forall x.\mathit{green}(x)$ is equivalent to $\exists x.\neg(\mathit{green}(x))$

Duality between logic operators

- De Morgan Law: for any predicate logic formulas φ_1 and φ_2

  $\neg(\varphi_1 \wedge \varphi_2)$ is equivalent to $\neg\varphi_1 \vee \neg\varphi_2$

  $\neg(\varphi_1 \vee \varphi_2)$ is equivalent to $\neg\varphi_1 \wedge \neg\varphi_2$

  $\neg\exists x.\varphi$ is equivalent to $\forall x.\neg\varphi$

  $\neg\forall x.\varphi$ is equivalent to $\exists x.\neg\varphi$

- Example: $\neg\forall x.\mathit{green}(x)$ is equivalent to $\exists x.\neg(\mathit{green}(x))$

- Corollary: for any relation symbol P and formula φ,

  $\neg\exists x.P(x) \wedge \varphi$ is equivalent to $\forall x.P(x) \rightarrow \neg\varphi$

  $\neg\forall x.P(x) \rightarrow \varphi$ is equivalent to $\exists x.P(x) \wedge \neg\varphi$

Examples

- (φ_1) There is no largest integer: $\neg\,(\exists y.\forall x.x \leq y)$
- (φ_2) For every integer one can find a larger integer: $\forall y.\exists x.x > y$

- Prove that φ_1 and φ_2 are equivalent

Examples

- (φ_1) There is no largest integer: $\neg\,(\exists y.\forall x.x \leq y)$
- (φ_2) For every integer one can find a larger integer: $\forall y.\exists x.x > y$

- Prove that φ_1 and φ_2 are equivalent

  $\neg\exists y.\forall x.x \leq y$ is equivalent to

  $\forall y.\neg\forall x.x \leq y$

Examples

- (φ_1) There is no largest integer: $\neg\,(\exists y.\forall x.x \leq y)$
- (φ_2) For every integer one can find a larger integer: $\forall y.\exists x.x > y$

- Prove that φ_1 and φ_2 are equivalent

  $\neg\exists y.\forall x.x \leq y$ is equivalent to

  $\forall y.\neg\forall x.x \leq y$ is equivalent to

  $\forall y.\exists x.\neg\ x \leq y$

Examples

- (φ_1) There is no largest integer: $\neg\,(\exists y.\forall x.x \leq y)$
- (φ_2) For every integer one can find a larger integer: $\forall y.\exists x.x > y$

- Prove that φ_1 and φ_2 are equivalent

  $\neg\exists y.\forall x.x \leq y$ is equivalent to

  $\forall y.\neg\forall x.x \leq y$ is equivalent to

  $\forall y.\exists x.\neg\ x \leq y$ is equivalent to

  $\forall y.\exists x.x > y$

Satisfiability, Validity and Equivalence

The Main Questions

- φ is satisfiable if there **exist** M and E that $M, E \models \varphi$
- φ is valid if **for all** M and E, $M, E \models \varphi$
- φ_1 is equivalent to φ_2 if, for all M and E:

  if $M, E \models \varphi_1$ then $M, E \models \varphi_2$, and viceversa

The Main Questions

- φ is satisfiable if there **exist** M and E that $M, E \models \varphi$
- φ is valid if **for all** M and E, $M, E \models \varphi$
- φ_1 is equivalent to φ_2 if, for all M and E:

$$\text{if } M, E \models \varphi_1 \text{ then } M, E \models \varphi_2, \text{ and viceversa}$$

Example: $\forall x.student(x) \rightarrow \exists y.instructor(y) \wedge younger(x, y)$

- is it satisfiable?
- is it valid?
- it is equivalent to:

$$\forall x.(\forall y.instructor(y) \rightarrow \neg younger(x, y)) \rightarrow \neg student(x)$$

How are these related?

- φ is satisfiable if there **exist** M and E that $M, E \models \varphi$

  i.e. it is not the case that: for all M and E, $M, E \models \neg\varphi$

  i.e. $\neg\varphi$ is **not** valid

- φ is valid if **for all** M and E, $M, E \models \varphi$

  i.e. $\neg\varphi$ is **not** satisfiable

- φ_1 is equivalent to φ_2 if, for all M and E:

  $$M, E \models \varphi_1 \text{ if, and only if, } M, E \models \varphi_2$$

  i.e. for all M and E: $M, E \models \varphi_1 \leftrightarrow \varphi_2$

  i.e. $\varphi_1 \leftrightarrow \varphi_2$ is valid

Using Satisfiability Checker to Answer Other Questions

- If we have an algorithm for checking satisfiability, then we can also check validity of formulas, and vice versa

```
bool VALID(formula F)
{ return (SAT(¬F) == false) }
```

Using Satisfiability Checker to Answer Other Questions

- If we have an algorithm for checking satisfiability, then we can also check validity of formulas, and vice versa

```
bool VALID(formula F)
{ return (SAT(¬F) == false) }
```

- If we have an algorithm for checking validity, then we can also check equivalence of formulas

```
bool EQUIV(formula F1, formula F2)
{ return VALID(F1↔F2) }
```

- Works for propositional logic as well as predicate logic

Complexity of Reasoning about Predicate Logic Formulas

- The problem of checking validity of predicate logic formulas is **undecidable**

  There is no algorithm that
  - for **all** predicate logic formulas
  - is guaranteed to **terminate** and
  - return the **correct** answer to the question of validity of its input formula

- What about satisfiability?

Complexity of Reasoning about Predicate Logic Formulas

- The problem of checking validity of predicate logic formulas is **undecidable**

  There is no algorithm that
  - for **all** predicate logic formulas
  - is guaranteed to **terminate** and
  - return the **correct** answer to the question of validity of its input formula

- What about satisfiability?

  Satisfiability is also undecidable (why?)

Complexity of Reasoning about Predicate Logic Formulas

- The problem of checking validity of predicate logic formulas is **undecidable**

  There is no algorithm that
  - for **all** predicate logic formulas
  - is guaranteed to **terminate** and
  - return the **correct** answer to the question of validity of its input formula

- What about satisfiability?

  Satisfiability is also undecidable (why?)

- Modern solvers are very effective in practice for checking validity and satisfiability of predicate logic formulas that we care about

Special Cases: Validity and Satisfiability of Predicate Logic

- Some restricted fragments of predicate logic are decidable
- Restrict the syntax
  - restrict quantification: for example, quantifier-free, $\exists^*\forall^*$, $\exists^*\forall\exists^*$
  - restrict the vocabulary: for example, monadic predicates
- Restrict the semantics

Special Cases: Validity and Satisfiability of Predicate Logic

- Some restricted fragments of predicate logic are decidable
- Restrict the syntax
  - restrict quantification: for example, quantifier-free, $\exists^*\forall^*$, $\exists^*\forall\exists^*$
  - restrict the vocabulary: for example, monadic predicates
- Restrict the semantics
  - Satisfiability Modulo Theories (SMT)
  - Only care about models that satisfy the axioms of some theory
  - Examples of theories: equality and uninterpreted functions, linear arithmetic over integers or rationals, arrays, bitvectors, etc.

Theory of Equality and Uninterpreted Functions (EUF)

- Vocabulary and intended interpretation
  - equality relation: designated binary relation symbol $=$ with intended meaning of equality, stipulated by *equality axioms*
  - any other constant, function, and relation symbols are uninterpreted (their meaning is not restricted by the axioms)

Theory of Equality and Uninterpreted Functions (EUF)

- Vocabulary and intended interpretation
  - equality relation: designated binary relation symbol $=$ with intended meaning of equality, stipulated by *equality axioms*
  - any other constant, function, and relation symbols are uninterpreted (their meaning is not restricted by the axioms)
- Example:
  - $\forall x.f(x) = x$ is satisfiable in EUF but not valid in EUF
  - $\forall x.\forall y.\ P(x) \wedge (f(y) = x) \rightarrow P(f(y))$ is valid in EUF

Theory of Equality and Uninterpreted Functions (EUF)

- Vocabulary and intended interpretation
  - equality relation: designated binary relation symbol $=$ with intended meaning of equality, stipulated by *equality axioms*
  - any other constant, function, and relation symbols are uninterpreted (their meaning is not restricted by the axioms)
- Example:
  - $\forall x.f(x) = x$ is satisfiable in EUF but not valid in EUF
  - $\forall x.\forall y.\ P(x) \wedge (f(y) = x) \rightarrow P(f(y))$ is valid in EUF
- Satisfiability in EUF
  - means there is a model and evaluation that satisfy both the formula and the axioms
  - is decidable for quantifier-free formulas
  - is undecidable in general

Axioms of EUF (for reference)

1. Reflexivity: $\forall x.x = x$
2. Symmetry: $\forall x, y.x = y \rightarrow y = x$
3. Transitivity: $\forall x, y, z.x = y \wedge y = z \rightarrow x = z$
4. Congruence: for every function symbol f of arity k,
   $\forall x_1, \ldots, x_k, y_1, \ldots, y_k.\ x_1 = y_1 \wedge \ldots \wedge x_k = y_k \rightarrow f(x_1, \ldots, x_k) = f(y_1 \ldots, y_k)$
5. Equivalence: for every relation symbol P of arity k,
   $\forall x_1, \ldots, x_k, y_1, \ldots, y_k.\ x_1 = y_1 \wedge \ldots \wedge x_k = y_k \rightarrow P(x_1, \ldots, x_k) \leftrightarrow P(y_1 \ldots, y_k)$

- Congruence and equivalence are axiom schemas

  i.e. they are a family of axioms containing one axiom for every function and relation symbol respectively

- For example, congruence for binary function f:

  $\forall x_1, x_2, y_1, y_2.\ x_1 = y_1 \wedge x_2 = y_2 \rightarrow f(x_1, x_2) = f(y_1, y_2)$

Arithmetics in Predicate Logic

Arithmetics in Predicate Logic

- Formalize the semantics of natural numbers in predicate logic
- Peano Arithmetic
  - addition and multiplication
  - undecidable
- Presburger Arithmetic
  - no multiplication
  - decidable
- Modular arithmetic
  - bounded numbers
  - addition and mutliplication
  - decidable
  - bitvector theory

Peano Arithmetic: Vocabulary and Intended Interpretation

- Vocabulary V consists of:
  - constant (i.e. nullary function) symbol *zero*
  - unary function symbol *succ*
  - binary function symbols *plus* and *mult*
  - binary relation symbol $=$
- The intended meaning of the symbols is:
  - *zero* represents the number 0
  - *succ* represents the successor function ($n \mapsto n + 1$)
  - *plus* and *mult* represent the usual addition and multiplication functions on natural numbers
  - equality represents natural number equality

Peano Arithmetic: Standard Model

- Universe U is the (infinite) set of natural numbers:

$$U = \mathbb{N} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots\}$$

- Interpretation maps symbols to their intended meaning:
  - $zero_M = \mathbf{0}$
  - $succ_M(n) = n + \mathbf{1}$, for every natural number n

    example: $succ_M(\mathbf{0}) = \mathbf{1}, succ_M(\mathbf{1}) = \mathbf{2}, \ldots$
  - $plus_M(n, m) = n + m$
  - $mult_M(n, m) = n * m$

    example: $\text{plus}_M(\mathbf{2}, \mathbf{3}) = \mathbf{5}$, $\text{mult}_M(\mathbf{0}, \mathbf{1}) = \mathbf{0}$

Peano Arithmetic: Shorthands

Intuitive notations for terms and formulas:

- Write the number k instead of k applications of *succ* to *zero*
  - 0 instead of the term *zero*, 1 instead of the term $succ(zero)$, 2 instead of $succ(succ(zero))$, etc.
- Write the number kx instead of summing k times x
  - $2x$ instead of the term $x + x$, $3x$ instead of the term $x + x + x$, etc.
- Write $+$ and $*$ instead of *plus* and *mult* and use infix notation
  - $x + y$ denotes the term $\text{plus}(x, y)$
  - $x * y$ denotes the term $\text{mult}(x, y)$

  For example: $2 = 1 + 1$ is a shorthand for:

$$succ(succ(zero)) = plus(succ(zero), succ(zero))$$

- Relations such as $<$ and $\geq$ are definable in predicate logic:

Peano Arithmetic: Shorthands

Intuitive notations for terms and formulas:

- Write the number k instead of k applications of *succ* to *zero*
  - 0 instead of the term *zero*, 1 instead of the term $succ(zero)$, 2 instead of $succ(succ(zero))$, etc.
- Write the number kx instead of summing k times x
  - $2x$ instead of the term $x + x$, $3x$ instead of the term $x + x + x$, etc.
- Write $+$ and $*$ instead of *plus* and *mult* and use infix notation
  - $x + y$ denotes the term $\text{plus}(x, y)$
  - $x * y$ denotes the term $\text{mult}(x, y)$

  For example: $2 = 1 + 1$ is a shorthand for:

$$succ(succ(zero)) = plus(succ(zero), succ(zero))$$

- Relations such as $<$ and $\geq$ are definable in predicate logic:
  - $x < y$ can be expressed as $\exists z.\neg(z = 0) \wedge x + z = y$

Peano Arithmetic: Examples

- $3x + 5 > 2y$ can be expressed as:

Peano Arithmetic: Examples

- $3x + 5 > 2y$ can be expressed as:

$$\exists z.\ \neg(z = 0) \wedge 3x + 5 = 2y + z$$

- How can we express the primality relation $\mathit{prime}(x)$?

Peano Arithmetic: Examples

- $3x + 5 > 2y$ can be expressed as:

$$\exists z.\ \neg(z = 0) \wedge 3x + 5 = 2y + z$$

- How can we express the primality relation $\mathit{prime}(x)$?

$$\forall y.(\exists z.x = y * z) \rightarrow y = 1 \vee y = x$$

- How about the theorem: "there are infinitely many prime numbers"?

Peano Arithmetic: Examples

- $3x + 5 > 2y$ can be expressed as:

$$\exists z.\ \neg(z = 0) \wedge 3x + 5 = 2y + z$$

- How can we express the primality relation $prime(x)$?

$$\forall y.(\exists z.x = y * z) \rightarrow y = 1 \vee y = x$$

- How about the theorem: "there are infinitely many prime numbers"?

$$\forall x.\exists y.\ y > x \wedge prime(y)$$

- Fermat's last theorem for $n = 3$:

$$\forall x, y, z.\ \neg(x * x * x + y * y * y = z * z * z)$$

- So, x^3 can be expressed by $x * x * x$. What about x^y?

Peano Arithmetic: Axioms

0. Axioms for equality
1. $\forall x.\ \neg(x + 1 = 0)$
   (zero is the first natural number, it has no predecessor)
2. $\forall x.\forall y.\ x + 1 = y + 1 \rightarrow x = y$
   (every number has at most one predecessor)
3. $\forall x.\ x + 0 = x$ (addition of 0)
4. $\forall x.\forall y.\ x + (y + 1) = (x + y) + 1$ (addition and successor)
5. $\forall x.\ x * 0 = 0$ (multiplication by 0)
6. $\forall x.\forall y.\ x * (y + 1) = x * y + x$ (multiplication and successor)
7. $(\varphi(0) \wedge (\forall x.\varphi(x) \rightarrow \varphi(x + 1))) \rightarrow \forall x.\varphi(x)$ (induction axiom schema)

Modular Arithmetic: Motivation

- Peano Arithmetic is not well defined for finite structures
- Consider a finite universe $U_n = \{0, 1, 2, \ldots, n-1\}$
- Arithmetic functions result is outside of U_n for some inputs in U_n
  - $U_8 = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots, \mathbf{7}\}$
  - successor of $\mathbf{7}$ is $\mathbf{8}$ which is not in U_8
  - result of $\mathbf{4} + \mathbf{5}$ is not in U_8
- So, we need to tweak the interpretation of arithmetic functions

Modular Arithmetic

- Successor, addition, and multiplication is computed modulo n:
  1. compute the interpretation as in (ordinary) arithmetic
  2. divide it by n
  3. return the remainder of the division by n
- For example:
  - the semantics of the term $4 + 5$ in U_8 is:

$$\mathbf{(4 + 5) \bmod 8 = 9 \bmod 8 = 1}$$

  - the semantics of the term $7 + 1$ in U_8 is $\mathbf{0}$

Modular Arithmetic

- Successor, addition, and multiplication is computed modulo n:
  1. compute the interpretation as in (ordinary) arithmetic
  2. divide it by n
  3. return the remainder of the division by n
- For example:
  - the semantics of the term $4 + 5$ in U_8 is:

$$(\mathbf{4 + 5}) \bmod \mathbf{8} = \mathbf{9} \bmod \mathbf{8} = \mathbf{1}$$

  - the semantics of the term $7 + 1$ in U_8 is $\mathbf{0}$
- The axioms of modular arithmetic are the same as PA apart from 1 (modified to the finite universe used) and 7 (not needed).

Arithmetic in Predicate Logic: Summary

- Peano Arithmetic is undecidable
- Presburger Arithmetic is decidable (i.e. Peano minus multiplication)
- Modular arithmetic is decidable

Predicate Logic vs Propositional Logic

Predicate Logic vs Propositional Logic: Satisfiability Checking

- Undecidable in predicate logic
- NP-Complete in propositional logic

Predicate Logic vs Propositional Logic: Satisfiability Checking

- Undecidable in predicate logic
- NP-Complete in propositional logic
- P-Complete in a special case of propositional logic called Horn Clauses

Special Case: Satisfiability of Horn Clauses

Syntax and semantics of Horn clauses:

- restricted propositional formulas
- clause is a disjunction of literals with only one positive literal
- example Horn clause: $v_1 \wedge v_2 \wedge v_3 \rightarrow v_4$

Special Case: Satisfiability of Horn Clauses

Syntax and semantics of Horn clauses:

- restricted propositional formulas
- clause is a disjunction of literals with only one positive literal
- example Horn clause: $v_1 \wedge v_2 \wedge v_3 \rightarrow v_4$

HORN-SAT problem is the problem of satisfiability of a given finite set of Horn Clauses

Special Case: Satisfiability of Horn Clauses

Syntax and semantics of Horn clauses:

- restricted propositional formulas
- clause is a disjunction of literals with only one positive literal
- example Horn clause: $v_1 \wedge v_2 \wedge v_3 \rightarrow v_4$

HORN-SAT problem is the problem of satisfiability of a given finite set of Horn Clauses

- HORN-SAT is P-complete
- linear time algorithm – a single call to unit propagation

Applications: relational databases, functional dependencies, SQL, transition systems, reachability, the shortest paths, XML, foreign keys

Tools for Automated Reasoning about Logical Formulas

- **SAT solvers**: tools for automating satisfiability checking in propositional logic
- **SMT Solvers**: tools for automating satisfiability checking in certain fragments and theories of predicate logic.
  - Modern SMT solvers support undecidable theories and quantifiers: effective at checking satisfiability of some formulas, but cannot in general provide guarantees on termination or completeness
- **Theorem provers**: tools for automating validity checking of predicate logic
- **Proof assistant**: tools for automating validity checking of higher-order logic
- Other tools for constraint solving and optimization

Competitions between Tools

- Competitions between different tools that support the same logic
  - SAT Solvers: Minisat, zChaff, Lingeling, glucose, HaifaSAT, ...
  - SMT Solvers: Z3, MathSat, veriT, CVC4, Yices, ...
  - Theorem provers: Vampire, SPASS, Paradox, iProver, ...
  - Proof Assistant: Isabelle, HOL, Coq, Satallax, ...
- Categories of problems within each logic
- Strong drive for performance improvement
- Community can submit benchmarks for competitions

Summary: How Hard is Satisfiability Checking?

- Horn Clauses: P-Complete
- Propositional Logic: NP-Complete
- Predicate Logic: undecidable
- Peano arithmetic: undecidable
- Presburger arithmetic: decidable
- Theory of equality and uninterpreted functions: undecidable
- Quantifier-free fragment of theory of equality and uninterpreted functions: decidable