Foundations of Data and Knowledge Bases

Preliminaries: Mathematical Notation

Joachim Niehren

Links: Linking Dynamic Data
Inria Lille

November 10, 2020

Outline

1. Functions and Relations
2. Kleene Star: Repetition
3. (First-Order) Terms
   - Abstract Syntax
   - Inductive Definitions
   - Arithmetic Expressions

Sets and Relations

Sets

- Booleans: $\mathbb{B} = \{0, 1\}$
- natural numbers: $\mathbb{N} = \{1, 2, \ldots\}$
- natural numbers of zero: $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$

Relations

for sets $A, B, A_1, \ldots, A_n$ where $n \in \mathbb{N}_0$ we define sets of:

- pairs: $A \times B = \{(a, b) \mid a \in A, b \in B\}$
- n-tuples: $A_1 \times \ldots \times A_n = \{(a_1, \ldots, a_n) \mid a_1 \in A_1, \ldots, a_n \in A_n\}$
  The set $A_1 \times \ldots \times A_n$ is called the type of such a tuple.
- subsets: $2^A = \{B \mid B \subseteq A\}$, the power set of A
- n-ary relations: $2^{A_1 \times \ldots \times A_n}$
  An n-ary relation is a table with n columns. The elements of column i belong to A_i for all $1 \leq i \leq n$.

Functions

- partial functions:

$$A \rightarrow_{partial} B = \{f \subseteq A \times B \mid \forall a \in A.\ \exists^{\leq 1} b \in B.\ (a, b) \in f\}$$

  For any partial function f we write $f(a) = b$ iff $(a, b) \in f$ and define the domain of f by:

$$dom(f) = \{a \in A \mid \exists b \in B. f(a) = b\}$$

- (total) functions: $A \rightarrow B = \{f \subseteq A \rightarrow_{partial} B \mid dom(f) = A\}$
  We write $f : A \rightarrow B$ instead of $f \in A \rightarrow B$

Exercise [Homework until next time]

1. How many elements has the function space $A \to B$ for two sets A and B? How many elements has $A \to \mathbb{B}$?
2. Let A be a set. Define a bijection $cf : 2^A \to (A \to \mathbb{B})$ that maps subsets B of A to their characteristic functions $cf(B)$.

Named Tuples

A named tuple is like a tuple except that its components are given names. Let N be a set of finite set of names and A another set.

Definition

A named tuple t with names in N and elements in A is a function $t : N \to A$.

If $N = \{n_1 \ldots, n_m\}$ with pairwise distinct n_i then we write:

$$t = [n_1/a_1, \ldots, n_m/a_m] \text{ iff } t(n_i) = a_i \text{ for all } n_i \in N$$

A named tuple is sometimes called a record, which are denoted as $t = \{n_1{:}a_1, \ldots, n_m{:}a_m\}$ in JSON, the Java Script Object Notation.

Typing of Named Tuples

A type of a named tuple states to which subset of A the elements of its components have to belong.

Typing

We say tat a named tuple t has type $\tau : N \to 2^A$ if $t(n) \in \tau(n)$ for all $n \in N$.

Exercise

How can you identify the set of n tuples in $A_1 \times \ldots \times A_n$ with some set of named tuples? Which names can you use? And which type? Provide a bijection between the set of n-tuples and your set of named tuples.

Named Relations

A named relation is like a relation, except that its columns are named.

Definition

Let N be a set of names and A a set. A named relation r with names in N and elements in A is a subset of name tuples with names in N and elements in A.

Typing

We say that a named relation r has type $\tau : N \to 2^A$ if r is a subsets of named tuples of type τ.

Exercise

How can you identify the set of relations in $2^{A_1 \times \ldots \times A_n}$ with some set of named relations? Which names can you use? And which type? Provide a bijection between the set of n-ary relations and your set of named n-ary relations?

Outline

1. Functions and Relations

2. Kleene Star: Repetition

3. (First-Order) Terms
   - Abstract Syntax
   - Inductive Definitions
   - Arithmetic Expressions

Transitive Closure

Let A be a set and $R \subseteq A \times A$ a binary relation on A. For all $i \in \mathbb{N}$ define:

Composition of Steps via R

$R^0 = \{(a, a) \mid a \in A\}$ 0 steps

$R^1 = R$ 1 step

$R^i = \{(a_1, a_3) \mid (a_1, a_2) \in R^{i-1}, (a_2, a_3) \in R\}$ i steps, where $i \in \mathbb{N}$

Iterating Steps via R

$R^+ = \cup_{i=1}^{\infty} R^i$ transitive closure

$R^* = R^0 \cup R^+$ reflexive transitive closure

Example

$R = \{(Lille, Paris), (Paris, Lyon), (Lyon, Marseille)\}$

$R^+ = R \cup \{(Lille, Lyon), (Lille, Marseille), (Paris, Marseille)\}$

$R^* = R^+ \cup \{(Lille, Lille), (Paris, Paris), (Lyon, Lyon), (Marseille, Marseille)\}$

Words

Alphabet

set of letters: Σ

Words in Σ^*

$$\Sigma^0 = \{\epsilon\}$$
$$\Sigma^i = \{w \cdot a \mid w \in \Sigma^{i-1}, a \in \Sigma\}$$
$$\Sigma^+ = \cup_{i=1}^{\infty} \Sigma^i$$
$$\Sigma^* = \Sigma^+ \cup \Sigma^0$$

How to define Concatenation?

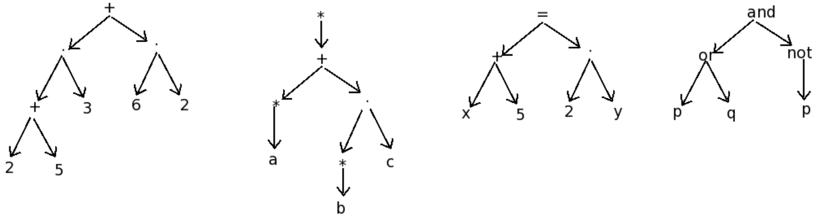
for all $w, w' \in \Sigma^*$ define $w \circ w' \in \Sigma^*$

Outline

1. Functions and Relations
2. Kleene Star: Repetition
3. (First-Order) Terms
   - Abstract Syntax
   - Inductive Definitions
   - Arithmetic Expressions

Examples for Terms

- terms are finite trees with fixed arities



- can be obtained by parsing concrete syntax of:
  - arithmetic expressions $(2 + 5) \cdot 3 + 6 \cdot 2$
  - equations $x + 5 = 2 \cdot y$
  - regular expressions $(a^* + b^* \cdot c)^*$
  - logic formulas $p \vee q \wedge \neg p$

Terms as Abstract Syntax

Concrete syntax

of an expression is a sequence of constants, operators, and parenthesis.

Abstract syntax

of an expression is a term

- obtained by from concrete syntax of the expression by parsing with respect to some grammar
- expresses the nesting structure of all operators, which is often induced by parenthesis or operator preferences
- ignores all details of concrete syntax, such as parenthesis and operator preferences.

Two Perspectives on Terms

Terms as nested structures

- basic values
- n-tuples of basic values
- n-tuples of m-tuples of basic values
- ...

Terms as graphs

- useful for graph algorithms
- more difficult for recursive algorithms along the term structure

Signature=Vocabulary

Ranked Signature $\Delta = (\Sigma, \mathrm{ar})$

- a set Σ of symbols
- a function $\mathrm{ar} : \Sigma \to \mathbb{N}_0$

Constants $a \in \Sigma$

are symbols with $\mathrm{ar}(a) = 0$

- basic values

Operators $f \in \Sigma$

are symbols of $\mathrm{ar}(f) \geq 1$

- constructors of tuples of values
- write $f(.,.)$ for operator of arity 2, etc.

Inductive Definition of Terms

Set of terms $Term_\Delta^{\leq m}$ of depth $\leq m$

$$
\begin{array}{rcl}
Term_\Delta^{\leq 0} & = & \{a \in \Sigma \mid \mathrm{ar}(a) = 0\} \\
Term_\Delta^{\leq m+1} & = & \{f(t_1, \ldots, t_n) \mid f \in \Sigma, \mathrm{ar}(f) = n,\ t_1, \ldots, t_n \in Term_\Delta^{\leq m}\} \\
 & \cup & Term_\Delta^{\leq m}
\end{array}
$$

Set of all terms

$$Term_\Delta \quad = \quad \cup_{m=0}^{\infty} Term_\Delta^{\leq m}$$

Recursive Definition of Terms

Set of terms $Term_\Delta$

is the least set that contains

- all constants $a \in \Sigma$ and
- all pairs $f(t_1, \ldots, t_n)$ consisting of an operator $f \in \Sigma$ of arity $\mathrm{ar}(f) = n$ and a tuple $(t_1, \ldots, t_n) \in (Term_\Delta)^n$

Notation

Mathematical

$$\begin{array}{rcl} \Sigma_n & = & \{f \in \Sigma \mid \mathrm{ar}(f) = n\} \\ \mathit{Term}_\Delta & = & \Sigma_0 \cup \cup_{n\geq 0} \Sigma_n \times (\mathit{Term}_\Delta)^n \end{array}$$

Backus-Naur form (BNF)

$t \in \mathit{Term}_\Delta ::= a \mid f(t_1, \ldots, t_n)$ where $n = \mathrm{ar}(f) > 0$ and $\mathrm{ar}(a) = 0$.

Arithmetic Expressions

Terms over signature

$\Delta = \mathbb{N} \cup \{+(.,.), \cdot(.,.)\}$ where all natural numbers are constants $\mathrm{ar}(n) = 0$.

Backus-Naur form

$t \in \mathit{Term}_\Delta ::= n \mid +(t_1, t_2) \mid \cdot(t_1, t_2)$ where $n \in \mathbb{N}$

Mathematical notation

$\mathit{Term}_\Delta \ = \ \mathbb{N} \ \cup \ \{\cdot, +\} \times (\mathit{Term}_\Delta \times \mathit{Term}_\Delta)$

Examples

- we identify $2 + 3 \cdot 5$ with term $+(2, \cdot(3, 5))$
- $2 + 3 \cdot 5$ is different from its value 17, but it can be evaluated to it.
- how to define an evaluator $\mathit{eval} : \mathit{Term}_\Delta \to \mathbb{N}$?

Evaluator

- define $eval : Term_\Delta \to \mathbb{N}$ by:

$$
\begin{array}{rcl}
eval(n) & = & n \\
eval(t_1 + t_2) & = & eval(t_1) + eval(t_2) \\
eval(t_1 \cdot t_2) & = & eval(t_1) \cdot eval(t_2)
\end{array}
$$

- for instance: $eval((2+3) \cdot 5) = 25$
- note that the symbols $+$ and $\cdot$ are overload; they are used as term constructors on the left and as the arithmetic functions on natural numbers on the right.
- the type of the function $eval$ resolves the ambiguity by overloading we could also annotate the function by its type to make the distinction, and write $+^{\mathbb{N}}$ resp. $\cdot^{\mathbb{N}}$ for instance.

Exercises [Homework]

3. Define the depth of an arithmetic term formally such that the depth of a constant is 0.
4. Define the number of nodes of an arithmetic term formally.

Equality

Structural equality on terms

We define structural equality $== \subseteq Term_\Sigma \times Term_\Sigma$ such that for all $f \in \Sigma$ with $\mathrm{ar}(f) = n$, terms $t, t_1, \ldots, t_n \in Term_\Sigma$ and constants $a \in \Sigma$:

a) $f(t_1, \ldots, t_n) == t$ iff t matches $f(t'_1, \ldots, t'_n)$ for some $t'_1, \ldots, t'_n$ such that $t_i = t'_i$ for all $1 \leq i \leq n$

b) $a == t$ if $t = a$ is the same constant of Σ

Node equality

Consider the term

$$t = f(g(a, b), f(g(a, b), a))$$

the subterms of t at nodes 1 and $2{\cdot}1$ are equal to $g(a, b)$ even though these two nodes are different.

Exercises

5. Compute the value of an arithmetic term in a programming language of your choice.

Exercises: Assignment 3 at UCC'2017

6. Reconsider arithmetic terms with the following abstract syntax:

   $$t \in T ::= 1 \mid +(t, t)$$

   Define a function $nl : T \to Nat$ in the language of mathematics such that $nl(t)$ is the number of leafs for any $t \in T$. Define the same function in the programming language Python.

7. Consider propositional formulas that have the following abstract syntax:

   $$f \in F ::= and(f, f) \mid or(f, f) \mid true \mid false$$

   Define a function $eval : F \to \mathbb{B}$ in the language of mathematics that evaluates formulas $f \in F$ to Booleans $eval(f)$. Define the same function in the programming language Python.

Exercises

8. Let Vars be a set. Consider propositional formulas with variables that have the following abstract syntax:

$$f \in F' ::= and(f, f) \mid or(f, f) \mid x = 1 \mid x = 0$$

where $x \in Vars$. Can you define *true* and *false* by equivalent formulas in F'?

9. For any $f \in F'$, let $V(f)$ be the set of variables that occur f. Define $V(f)$ formally in the language of mathematics, and also in Python.

10. Define a function $eval' : F' \times (\mathrm{Vars} \to \mathbb{B}) \to \mathbb{B}$ in the language of mathematics, that evaluates any formula $f \in F'$ to a Boolean $eval'(f, \alpha)$ when given a variable assignment $\alpha : \mathrm{Vars} \to \mathbb{B}$ as second input argument. Define the same function in the programming language Python.

Exercises

11. We call a formula $f \in F$ satisfiable, if there exists a variable assignment that makes f true, i.e., if there exists $\alpha : \mathrm{Vars} \to \mathbb{B}$ with $eval'(f, \alpha) = 1$. Let $sat : F' \to \mathbb{B}$ be the function such that $sat(f) = 1$ iff f is satisfiable. Can you define the function sat in the Python? You have to find an algorithm that the computes sat, and implement your algorithm in Python. What is the worst case running time of your algorithm?
12. Which is the most famous problem that is known to be NP-complete? How is it related to the above decision problem sat?

Exercises

13. Consider the relational database with two elements $D = \{0, 1\}$ and two monadic relations, $Zero = \{0\}$ and $One = \{1\}$. It can be queried by formulas ϕ with the following abstract syntax where x ranges over variable in a set Vars:

$$\phi ::= Zero(x) \mid One(x) \mid \phi \wedge \phi' \mid \phi \vee \phi'$$

An answer of a database query ϕ is a variable assignment $\alpha : \mathrm{Vars} \to D$ that makes ϕ true on D. How difficult is it to decide whether a query ϕ has an answer for the above database?