The Knotty Companion

Vu Phan

2017/01/06

Abstract

This document is the specification of the `Knotty` language.

Contents

1 Syntax

1.1 Lexicon

These are **reserved lexemes**:

```
unknown constant function let return check
, ( ) :=
if else
true false
```

These are **operators**:

```
or and not
= /= > < >= <=
+ - * / % ^
```

A **number** is either:

- `i`, or
- one or more consecutive digits (`0`-`9`)

An **identifier** is one letter (`a`-`z` or `A`-`Z`) followed by zero or more letters and digits. Also, an identifier must not be a reserved lexeme, an operator, or a number.

Note: blank characters (spaces, tabs, new-lines) are delimiters.

1.2 Grammar

1.2.1 Names

An **unknown name** is an identifier. So is a **constant name**, a **function name**, a **temporary name**, and a **check name**.

1.2.2 Named Terms

A **named term** is an unknown name, a constant name, a function name, a formal parameter, a temporary name, or an actual function expression.

1.2.3 Parameters

A **formal parameter** is an identifier. An **actual parameter** is a term.

1.2.4 Function Expressions

A **formal function expression** has the form

$f\,(p_1\,, \ldots,\; p_n)$

where f is a function name and each p_i is a formal parameter ($n > 0$).

An **actual function expression** has the same form, but with each p_i being an actual parameter.

1.2.5 Arithmetic Terms

An **inner arithmetic term**:

- is a number, or
- is a named term, or
- has the form

  `(`t`)`

  where t is an inner arithmetic term.

An **arithmetic term**:

- is an inner arithmetic term, or
- has the form

  `-`t

  where t is an arithmetic term, or
- has the form

  $t_1 \diamond t_2$

  where t_1 & t_2 are terms and $\diamond \in \{+, -, *, /, \%, \hat{}\}$.

1.2.6 Boolean Terms

A **comparison boolean term** has the form

$t_1 \diamond t_2$

where t_1 & t_2 are arithmetic terms and $\diamond \in \{=, / =, >, <, >=, <=\}$.

An **inner boolean term**:

- is the keyword `true`, or
- is the keyword `false`, or
- is a named term, or
- has the form

  `(`t`)`

  where t is an inner boolean term.

A **boolean term**:

- is a comparison boolean term, or
- is an inner boolean term, or
- has the form

  `not` t

  where t is a boolean term, or
- has the form

  $t_1 \diamond t_2$

  where t_1 & t_2 are boolean terms and $\diamond \in \{\texttt{or}, \texttt{and}\}$.

1.2.7 Terms

An **inner term**:

- is a named term, or
- has the form

  `(`t`)`

  where t is an inner term.

A **conditional term** has the form

t_1 `if` t_2 `else` t_3

where t_1 & t_3 are terms and t_2 is a boolean term.

A **term** is an inner term, an arithmetic term, a boolean term, or a conditional term.

1.2.8 Unknown Statements

An **unknown statement** has the form

`unknown` u_1`,`$\ldots$`,` u_n

where each u_i is an unknown name ($n > 0$).

1.2.9 Constant Statements

A **constant statement** has the form

`constant` c `:=` t

where c is a constant name and t is a term.

1.2.10 Function Statements

A **return clause** has the form

`return` t

where t is a term.

A **let clause** has the form

`let` tmp `:=` t

where tmp is a temporary name and t is a term.

A **function statement** has the form

`function` fe

lc_1

$\ldots$

lc_n

rc

where fe is a formal function expression, each lc_i is a let clause ($n \geq 0$), and rc is a return clause.

1.2.11 Check Statements

A **check statement** has the form

```
check c := t
```

where c is a check name and t is a term.

1.2.12 Program

A **program** has the form

s_1
$\dots$
s_n

where each s_i is an unknown statement, a constant statement, a function statement, or a check statement ($n \geq 0$).

2 Semantics

2.1 Operations

Operator	Meaning
`or`	disjunction
`and`	conjunction
`not`	negation
=	equal
/=	unequal
>	greater
<	less
>=	greater or equal
<=	less or equal
+	plus
-	unary or binary minus
*	multiplication
/	division
%	modulo
ˆ	exponentiation

Note: parentheses override usual operator precedence.

2.2 Namespace

Name	Scope
unknown	program
constant	program
function	program
temporary	function

2.3 Values

The number `i` is the imaginary unit.

An unknown name represents a complex number of unspecified value.

A constant name represents the term in the corresponding constant statement.

A function name represents the mapping defined in the corresponding function statement.

A temporary name represents the term in the corresponding temporary clause.

A check name represents the term in the corresponding check statement.

2.4 Input/Output

The `Knotty Engine`:

- accepts a `Knotty` program
- generates a TEX program showing the check names and their corresponding values as specified by the `Knotty` program.