LogicGuard Type System*

Bashar Ahmad Michael Krieger

RISC Software GmbH, Unit Advanced Computing Technologies

{bashar.ahmad, michael.krieger}@risc-software.at

Abstract

LogicGuard is a framework used to develop formalised specifications to verify and monitor network activities at runtime using predicate logic. This report describes the type system developed for the LogicGuard specification language. The LogicGuard specification language is a typed language and is statically type checked. Based on the language specification we derived the type system's judgments and rules, and accordingly the type checker was implemented.

1 Introduction

The aim of type system is to prevent runtime execution errors. So, when a language holds such a property for all possible programs written in that language then it is considered to be **type sound**. In order to design such a type system precise notations and definitions need to be developed, and detailed proof of formal properties of the definitions need to be provided. A **typed language** means, that all variables defined in such language are given types (a restrict range of possible values). For example a variable x of type `boolean` can only assume boolean values during every execution of the program. Consequently, untyped languages don't have types or have a single universal type for all variables. Missing type information means that an operation may be applied to inappropriate variables which may result in corrupted data, execution errors or unknown behaviour. A type system keeps track of all variables and their type information in a program and is used to determine whether a program is well behaved or not. A typed language can enforce good behaviour by statically type checking the program before execution. This process is called **type checking** and the algorithm used is called **type checker**. So when a program passes the type checker it is said to be **well typed**, otherwise it is **ill typed**. Static checking cannot guarantee the prevention of all kind of execution errors, so dynamic checking has to be performed dynamically. For example array bounds must be checked dynamically to prevent out of bound errors.

The purpose of LogicGuard was to build a framework which is able to monitor network activities in runtime using predicate logic. In order to accomplish this goal a specification language was developed. This specification language describes only the predicate logic formulas, monitor, terms (stream, value and position) and a set of fixed functions. This means that the specification language is abstracted from the low level language which is used for computation and handling the actual networking functions. This separation is intentional, since the focus of the project was to be able to specify the logic of the monitors in an abstract and optimised way. The computation and the handling of network functions is done using existing programming languages and their libraries which are already optimised.

The developed type system has to handle two type sets:

*The project "LogicGuard: The Efficient Checking of Time-Quantified Logic Formulas with Applications in Computer Security" is sponsored by the FFG BRIDGE program, project No. 832207.

- first the internal types which are `monitor`,`formula` and `term`. `Term` is further divided into three sub types, `stream` term, `value` term and `position` term.
- second the external types `int, float, ip, etc.`.

The LogicGuard specification has no knowledge or control over external types, therefore it is required within the specification to define any external type being used, so the type checker will be able to check unknown types against these definitions and will keep track of type information. It is also required to provide definitions of external functions and streams. These means provide explicit type information required for the arguments and the return values, so the type checker could check against these definitions whenever an external function is called.

As mentioned earlier for a language to be classified as **type sound** language it has to hold a certain property, that is the type system has to guarantee that no execution error could occur during runtime for all possible programs in such a language, To prove such claim a very detailed analysis and study of the type system has to be done, however it is not necessary to apply all the formal techniques in full for a type system to be useful. By applying basic type system principles we can avoid a fair number of obvious and non-obvious execution errors. For now we cannot claim the soundness of the type system for the LogicGuard specification language, however, through our usage and practical experience with our type system we have been able to prevent a fair amount of execution errors which satisfies our target for the time being.

The rest of report is structured as the following: section 2 shows the definitions used in the type system, section 3 describes the judgments, section 4 derive rules and in section 5 conclude the report. An appendix is attached to the report which shows the language specification.

2 Definitions

2.1 Abbreviations

```
E: Environment
I: Identifier information.
T: Type information.
V: Term value.
D: Declaration
```

2.2 Definitions

```
Non-Terminal:
- Specification
- Statement
- Formula
- Term
- PositionTerm
- ValueTerm
- StreamTerm
- Bind
- Constraint
- FunctionCall
- Parameter
- ExtType
- ExtImportDec
- ExtFuncDec
```

```
- ExtPredicateDec
- ExtStreamDec
- BinaryFunction
- Range
- FormulaDec
- PositionDec
- StreamDecl
- ValueDecl
- ParamDecl
- Monitor
- MonitorPositionDec
- Position
- Type

Terminal:
- ID
- STRING
- INT
- FLOAT
- Not = ~
- Op = &&, ||, => , <=>
- Rel = >, >=, <, >=
- Sign + , -

E = Id -> (I, Decl)

T = ID

V = |position
|(T)value
|(T)stream

I = (V*, T) ValueTermInfo
|(V*, T) StreamTermInfo
|(V*) PositionTermInfo
|(V*) FormulaInfo
|MonitorInfo
|(V*, T)ExtFuncDecInfo
|(V*, T)ExtStreamDecInfo
|(V*)ExtPredicateDecInfo

Decl = FD(ID, V*) | SD(ID, V*) | PD(ID, V*) | VD(ID, V*)
PosId = (Id, Id)
```

3 Judgments

$\Phi \vdash$ Specification : (E)specification

E $\vdash$ StatementSeq : (E)statementSeq

E $\vdash$ Statement : (E)statement

E $\vdash$ Formula : formula

E $\vdash$ PositionTerm : positionTerm

$\mathtt{E} \vdash \mathtt{ValueTerm} : (\mathtt{T})\mathtt{valueTerm}$

$\mathtt{E} \vdash \mathtt{StreamTerm} : (\mathtt{T})\mathtt{streamTerm}$

$\mathtt{E} \vdash \mathtt{ExtFuncDec} : (\mathtt{E})\mathtt{extFuncDecl}$

$\mathtt{E} \vdash \mathtt{ExtPredicateDec} : (\mathtt{E})\mathtt{extPredicateDecl}$

$\mathtt{E} \vdash \mathtt{ExtStreamDec} : (\mathtt{E})\mathtt{extStreamDecl}$

$\mathtt{E} \vdash \mathtt{FormulaDec} : (\mathtt{E})\mathtt{formulaDecl}$

$\mathtt{E} \vdash \mathtt{PositionDec} : (\mathtt{E})\mathtt{positionDecl}$

$\mathtt{E} \vdash \mathtt{StreamDecl} : (\mathtt{E})\mathtt{streamDecl}$

$\mathtt{E} \vdash \mathtt{ValueDecl} : (\mathtt{E})\mathtt{valueDecl}$

$\mathtt{E} \vdash \mathtt{Term} : (\mathtt{V})\mathtt{term}$

$\mathtt{E} \vdash \mathtt{Bind} : (\mathtt{E})\mathtt{bind}$

$\mathtt{E} \vdash \mathtt{Constraint} : \mathtt{constraint}$

$\mathtt{E} \vdash \mathtt{FunctionCall} : (\mathtt{T})\mathtt{functionCall}$

$\mathtt{E} \vdash \mathtt{ParameterDec} : (\mathtt{E})\mathtt{parameterDec}$

$\mathtt{E} \vdash \mathtt{ExtTypeDec} : (\mathtt{E})\mathtt{extTypeDec}$

$\mathtt{E} \vdash \mathtt{ExtImportDec} : \mathtt{extImportDec}$

$\mathtt{E} \vdash \mathtt{BinaryFunction} : (\mathtt{V})\mathtt{binaryFunction}$

$\mathtt{E} \vdash \mathtt{Range} : (\mathtt{ID})\mathtt{range}$

$\mathtt{E} \vdash \mathtt{ParamDecl} : (\mathtt{V})\mathtt{paramDecl}$

$\mathtt{E} \vdash \mathtt{Monitor} : \mathtt{monitor}$

$\mathtt{E} \vdash \mathtt{MonitorPositionDec} : (\mathtt{ID})\mathtt{monitorPositionDec}$

$\mathtt{E} \vdash \mathtt{Position} : \mathtt{position}$

$\mathtt{E} \vdash \mathtt{Type} : (\mathtt{T})\mathtt{type}$

$\mathtt{E} \vdash \mathtt{FunctionRef} : (\mathtt{ID})\mathtt{functionRef}$

4 Rules

Specification:

$$\frac{\Phi \vdash \mathtt{StatementSeq} : (\mathtt{E})\mathtt{statementSeq}}{\vdash \mathtt{StatementSeq} : (\mathtt{E})\mathtt{specification}}$$

StatementSeq:

$\mathtt{E} \vdash \epsilon : (\Phi)\mathtt{statementSeq}$

$$\frac{(\mathtt{E0}) \vdash \mathtt{Statement} : (\mathtt{E2})\mathtt{statement} \quad (\mathtt{E0} + \mathtt{E2}) \vdash \mathtt{StatementSeq} : (\mathtt{E4})\mathtt{statementSeq}(\mathtt{E1} = \mathtt{E2} + \mathtt{E4})}{(\mathtt{E0}) \vdash \mathtt{Statement}; \mathtt{StatementSeq} : (\mathtt{E1})\mathtt{statementSeq}}$$

Statement:

$$\frac{(\texttt{E}) \vdash \texttt{Monitor : monitor}}{(\texttt{E}) \vdash \texttt{Monitor} : (\Phi)\texttt{Statement}}$$

$$\frac{(\texttt{E0}) \vdash \texttt{Bind} : (\texttt{E1})\texttt{bind}}{(\texttt{E0}) \vdash \texttt{Bind} : (\texttt{E1})\texttt{statement}}$$

$$\frac{(\texttt{E}) \vdash \texttt{ExtImportDec} : \texttt{extImportDec}}{(\texttt{E}) \vdash \texttt{ExtImportDec} : \Phi\texttt{statement}}$$

$$\frac{(\texttt{E0}) \vdash \texttt{ExtTypeDec} : (\texttt{E1})\texttt{extTypeDec}}{(\texttt{E0}) \vdash \texttt{ExtTypeDec} : (\texttt{E1})\texttt{statement}}$$

$$\frac{(\texttt{E0}) \vdash \texttt{ExtFunctionDec} : (\texttt{E1})\texttt{extFunctionDec}}{(\texttt{E0}) \vdash \texttt{ExtFunctionDec} : (\texttt{E1})\texttt{statement}}$$

$$\frac{(\texttt{E0}) \vdash \texttt{ExtPredicateDec} : (\texttt{E1})\texttt{extPredicateDec}}{(\texttt{E0}) \vdash \texttt{ExtPredicateDec} : (\texttt{E1})\texttt{statement}}$$

$$\frac{(\texttt{E0}) \vdash \texttt{ExtStreamDec} : (\texttt{E1})\texttt{extStreamDec}}{(\texttt{E0}) \vdash \texttt{ExtStreamDec} : (\texttt{E1})\texttt{statement}}$$

Monitor

$$\frac{\begin{array}{l}(\texttt{E}) \vdash \texttt{MonitorPositionDecSeq} : (\texttt{E0})\texttt{monitorPositionDecSeq} \\ (\texttt{E} + \texttt{E0}) \vdash \texttt{Formula} : \texttt{formula}\end{array}}{(\texttt{E}) \vdash \texttt{ID}; \texttt{MonitorPositionDecSeq}; \texttt{Formula} : (\Phi)\texttt{monitor}}$$

MonitorPositionDecSeq

$$(\texttt{E}) \vdash \Phi : \texttt{monitorPositionDecSeq}$$

$$\frac{\begin{array}{l}(\texttt{E}) \vdash \texttt{MonitorPositionDec} : \texttt{E2monitorPositionDec}(\texttt{E} + \texttt{E2}) \vdash \texttt{MonitorPositionDecSeq} : \\ (\texttt{E3})\texttt{monitorPositionDecSeq}(\texttt{E1} = \texttt{E2} + \texttt{E4})\end{array}}{(\texttt{E}) \vdash \texttt{MonitorPositionDec}; \texttt{MonitorPositionDecSeq} : (\texttt{E1})\texttt{monitorPositionDecSeq}}$$

MonitorPositionDec

$$(\texttt{E}) \vdash \texttt{MonitorPositionDec} : (\texttt{E1})\texttt{monitorPositionDec}$$

Formula

$$(\mathtt{E}) \vdash \mathtt{ID} : \mathtt{formula}$$

$$(\mathtt{E}) \vdash \mathtt{true} : \mathtt{formula}$$

$$(\mathtt{E}) \vdash \mathtt{false} : \mathtt{formula}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{Formula} : \mathtt{formula}}{(\mathtt{E}) \vdash \ \mathtt{Formula} : \mathtt{formula}}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{Formula1} : \mathtt{formula} (\mathtt{E}) \vdash \mathtt{Formula2} : \mathtt{formula}}{(\mathtt{E}) \vdash \mathtt{Formula1}; \mathtt{Op}; \mathtt{Formula2} : \mathtt{formula}}$$

$$\frac{(\mathtt{E0}) \vdash \mathtt{Bind} : (\mathtt{E2})\mathtt{bind}(\mathtt{E1} = \mathtt{E0} + \mathtt{E2})(\mathtt{E1})\mathtt{Formula} : \mathtt{formula}}{(\mathtt{E0}) \vdash \mathtt{Bind}; \mathtt{Formula} : \mathtt{formula}}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{TermSeq} : (\mathtt{V}*)\mathtt{termSeq}}{(\mathtt{E}) \vdash \mathtt{ID}; \mathtt{TermSeq} : \mathtt{formula}} \ \mathtt{match}(\mathtt{V}*, \mathtt{T}*)$$

$$\frac{\begin{array}{l}(\mathtt{E}) \vdash \mathtt{StreamTerm} : (\mathtt{T})\mathtt{streamTerm} \\ (\mathtt{id}) \vdash \mathtt{Range} : \mathtt{range}(\mathtt{E} + \mathtt{id}) \vdash \mathtt{Formula} : \mathtt{formula} \ \ (\mathtt{ID} == \mathtt{id})\end{array}}{(\mathtt{E})\mathtt{ID}; \mathtt{StreamTerm}; \mathtt{Range}; \mathtt{Formula} : \mathtt{formula}}$$

TypeSeq

$$(\mathtt{E}) \vdash \Phi : \mathtt{typeSeq}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{Type} : (\mathtt{T})\mathtt{type} (\mathtt{E}) \vdash \mathtt{TypeSeq} : (\mathtt{TS})\mathtt{TypeSeq}}{(\mathtt{E})\mathtt{Type}; \mathtt{TypeSeq} : (\mathtt{T} + \mathtt{TS})\mathtt{TypeSeq}}$$

$$\mathtt{TS} = \mathtt{T}*$$

Type

$$(\mathtt{E}) \vdash \mathtt{ID} : (\mathtt{T} - > \mathtt{ID})\mathtt{type}$$

ExtTypeDec

$$(\mathtt{E}) \vdash \mathtt{ID} : (\mathtt{ID} - > \mathtt{ExtTypeDec}(\mathtt{ID}))\mathtt{extTypeDec}$$

ExtImportDec

$$\mathtt{E} \vdash \mathtt{ID} : \mathtt{extImportDec}$$

ExtStreamDec

$$\frac{(\mathtt{E}) \vdash \mathtt{Type} : (\mathtt{T})\mathtt{type}}{(\mathtt{E}) \vdash \mathtt{Type}; \mathtt{ID} : (\mathtt{ID} -> \mathtt{extStreamDec}(\mathtt{T}, \mathtt{ID}))\mathtt{extStreamDec}}$$

ExtFunctionDec

$$\frac{(\mathtt{E}) \vdash \mathtt{Type} : (\mathtt{T})\mathtt{type} (\mathtt{E}) \vdash \mathtt{TypeSeq} : (\mathtt{TS})\mathtt{typeSeq}}{(\mathtt{E}) \vdash \mathtt{Type}; \mathtt{ID}; \mathtt{TypeSeq} : (\mathtt{id} -> \mathtt{ExtFunctionDec}(\mathtt{ID}, \mathtt{T}, \mathtt{TS}))\mathtt{extFunctionDec}}$$

ExtPredicateDec

$$\frac{(\mathtt{E}) \vdash \mathtt{TypeSeq} : (\mathtt{TS})\mathtt{typeSeq}}{(\mathtt{E}) \vdash \mathtt{ID}; \mathtt{typeSeq} : (\mathtt{id} -> \mathtt{ExtPredicateDec}(\mathtt{ID}, \mathtt{TS}))\mathtt{extPredicateDec}}$$

Bind

$$\frac{(\mathtt{E}) \vdash \mathtt{FormulaDec} : (\mathtt{FD})\mathtt{formulaDec} (\mathtt{E} + \mathtt{FD}) \vdash \mathtt{Formula} : \mathtt{formula}}{(\mathtt{E}) \vdash \mathtt{FormulaDec} = \mathtt{Formula} : (\mathtt{E1} = \mathtt{I} -> \mathtt{FD})\mathtt{bind}}$$

$$\mathtt{FD} : \mathtt{FormulaDeclaration}(\mathtt{ID}, \mathtt{V}*)$$

$$\frac{(\mathtt{E0}) \vdash \mathtt{StreamDec} : (\mathtt{SD})\mathtt{streamDec} (\mathtt{E0} + \mathtt{SD}) \vdash \mathtt{StreamTerm} : \mathtt{streamTerm}}{(\mathtt{E0}) \vdash \mathtt{StreamDec} = \mathtt{StreamTerm} : (\mathtt{E1} = \mathtt{I} -> \mathtt{SD})\mathtt{bind}}$$

$$\mathtt{SD} : \mathtt{StreamDeclaration}(\mathtt{ID}, \mathtt{V}*)$$

$$\frac{(\mathtt{E0}) \vdash \mathtt{ValueDec} : (\mathtt{VD})\mathtt{valueDec} (\mathtt{E0} + \mathtt{VD}) \vdash \mathtt{ValueTerm} : \mathtt{valueTerm}}{(\mathtt{E0}) \vdash \mathtt{ValueDec} = \mathtt{ValueTerm} : (\mathtt{E1} = \mathtt{I} -> \mathtt{VD})\mathtt{bind}}$$

$$\mathtt{VD} : \mathtt{ValueDeclaration}(\mathtt{ID}, \mathtt{V}*)$$

$$\frac{(\mathtt{E0}) \vdash \mathtt{PositionDec} : (\mathtt{PD})\mathtt{positionDec} (\mathtt{E0} + \mathtt{PD}) \vdash \mathtt{PositionTerm} : \mathtt{positionTerm}}{(\mathtt{E0}) \vdash \mathtt{PositionDec} = \mathtt{PositionTerm} : (\mathtt{E1} = \mathtt{I} -> \mathtt{PD})\mathtt{bind}}$$

$$\mathtt{PD} : \mathtt{PositionDeclaration}(\mathtt{ID}, \mathtt{V}*)$$

FormulaDec

$$\frac{(\mathtt{E0}) \vdash \mathtt{ParameterDecSeq} : (\mathtt{V}*)\mathtt{parameterDecSeq}}{(\mathtt{E0}) \vdash \mathtt{ID}; \mathtt{ParameterDecSeq} : (\mathtt{FD})\mathtt{formulaDec}}$$

PositionDec

$$\frac{(\mathtt{E0}) \vdash \mathtt{ParameterDecSeq} : (\mathtt{V}*)\mathtt{parameterDecSeq}}{(\mathtt{E0}) \vdash \mathtt{ID}; \mathtt{ParameterDecSeq} : (\mathtt{PD})\mathtt{positionDec}}$$

StreamDec

$$\frac{(\texttt{E0}) \vdash \texttt{ParameterDecSeq} : (\texttt{V*})\texttt{parameterDecSeq}}{(\texttt{E0}) \vdash \texttt{ID}; \texttt{ParameterDecSeq} : (\texttt{SD})\texttt{streamDec}}$$

ValueDec

$$\frac{(\texttt{E0}) \vdash \texttt{ParameterDecSeq} : (\texttt{V*})\texttt{parameterDecSeq}}{(\texttt{E0}) \vdash \texttt{ID}; \texttt{ParameterDecSeq} : (\texttt{VD})\texttt{valueDec}}$$

Range

$$\frac{(\texttt{E}) \vdash \texttt{positionTerm} : \texttt{positionTerm}}{(\texttt{E}) \vdash \texttt{positionTerm}; \texttt{REL}; \texttt{ID} : (\texttt{ID})\texttt{range}}$$

$$\frac{(\texttt{E}) \vdash \texttt{PositionTerm1} : \texttt{positionTerm} (\texttt{E}) \vdash \texttt{PositionTerm2} : \texttt{positionTerm}}{(\texttt{E}) \vdash \texttt{PositionTerm1}; \texttt{REL}; \texttt{ID}; \texttt{REL}; \texttt{PositionTerm2} : (\texttt{ID})\texttt{range}}$$

ParameterDecSeq

$$(\texttt{E}) \vdash \Phi : (\Phi)\texttt{ParameterDecSeq}$$

$$\frac{(\texttt{E}) \vdash \texttt{ParameterDec} : (\texttt{V})\texttt{parameterDec} (\texttt{V*}) \vdash \texttt{ParameterDecSeq} : \texttt{parameterDecSeq}}{(\texttt{E}) \vdash \texttt{ParameterDec}; \texttt{ParameterDecSeq} : (\texttt{V*})\texttt{parameterDecSeq}}$$

ParameterDec

$$\frac{(\texttt{E}) \vdash \texttt{FormulaParameterDec} : \texttt{formulaParameterDec}}{(\texttt{E0}) \vdash \texttt{FormulaParameterDec} : (\texttt{V})\texttt{parameterDec}}$$

$$\frac{(\texttt{E0}) \vdash \texttt{PositionParameterDec} : \texttt{positionParameterDec}}{(\texttt{E0}) \vdash \texttt{PositionParameterDec} : (\texttt{V})\texttt{parameterDec}}$$

$$\frac{(\texttt{E0}) \vdash \texttt{StreamParameterDec} : (\texttt{T})\texttt{streamParameterDec}}{(\texttt{E0}) \vdash \texttt{StreamParameterDec} : (\texttt{V})\texttt{parameterDec}}$$

$$\frac{(\texttt{E0}) \vdash \texttt{ValueParameterDec} : (\texttt{T})\texttt{ValueParameterDec}}{(\texttt{E0}) \vdash \texttt{ValueParameterDec} : (\texttt{V})\texttt{parameterDec}}$$

FormulaParameterDec

$$(\texttt{E}) \vdash \texttt{FormulaParameterDec} : \texttt{formulaParameterDec}$$

PositionParamDec

$$(\texttt{E}) \vdash \texttt{PositionParameterDec} : \texttt{positionParameterDec}$$

ValueParameterDec

$$(\texttt{E}) \vdash \texttt{Type}; \texttt{ID} : (\texttt{T})\texttt{valueParameterDec}$$

StreamParameter

$$(\mathtt{E}) \vdash \mathtt{Type}; \mathtt{ID} : (\mathtt{T})\mathtt{streamParameter}$$

TermSeq

$$(\mathtt{E0}) \vdash \Phi : (\mathtt{PHI})\mathtt{termseq}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{Term} : (\mathtt{V})\mathtt{term} (\mathtt{E}) \vdash \mathtt{TermSeq} : (\mathtt{V}*)\mathtt{termseq}}{(\mathtt{E})\mathtt{term}; \mathtt{termseq} : (\mathtt{V}*)\mathtt{termseq}}$$

Term

$$\frac{(\mathtt{E}) \vdash \mathtt{SteamTerm} : (\mathtt{T})\mathtt{steamterm}}{(\mathtt{E}) \vdash \mathtt{SteamTerm} : (\mathtt{V})\mathtt{term}}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{Valueterm} : (\mathtt{T})\mathtt{valueterm}}{(\mathtt{E}) \vdash \mathtt{ValueTerm} : (\mathtt{V})\mathtt{term}}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{positionterm} : \mathtt{positionterm}}{(\mathtt{E}) \vdash \mathtt{positionterm} : (\mathtt{V})\mathtt{term}}$$

PostionTerm

$$(\mathtt{E}) \vdash \mathtt{ID} : \mathtt{positionTerm}$$

$$(\mathtt{E}) \vdash 0 : \mathtt{positionTerm}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{PositionTerm} : \mathtt{positionterm}}{(\mathtt{E}) \vdash \mathtt{PositionTerm}(+/-)\mathtt{INT} : \mathtt{positionterm}}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{StreamTerm} : (\mathtt{T})\mathtt{streamterm} (\mathtt{E}) \vdash \mathtt{range} : (\mathtt{ID1})\mathtt{range} (\mathtt{E}) \vdash \mathtt{formula} : \mathtt{formula}}{(\mathtt{E0}) \vdash \mathtt{ID}; \mathtt{streamterm}; \mathtt{range}; \mathtt{formula} : \mathtt{positiontermifmatch}(\mathtt{ID1}, \mathtt{ID})}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{Bind} : (\mathtt{E1})\mathtt{bind}(\mathtt{E} + \mathtt{E1}) \vdash \mathtt{PositionTerm} : \mathtt{positionterm}}{(\mathtt{E}) \vdash \mathtt{Bind}; \mathtt{PositionTerm} : \mathtt{positionterm}}$$

ValueTerm

$$(\mathtt{E}) \vdash \mathtt{ID} : \mathtt{valueterm}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{Range} : (\mathtt{ID})\mathtt{Range}}{(\mathtt{E}) \vdash \mathtt{Range} : \mathtt{valueterm}}$$

$$\frac{(\mathtt{E}) \vdash \mathtt{StreamTerm} : (\mathtt{T1})\mathtt{streamterm} (\mathtt{E}) \vdash \mathtt{Range} : (\mathtt{ID})\mathtt{range} (\mathtt{E}) \vdash \mathtt{Formula} : \mathtt{formula}}{(\mathtt{E}) \vdash \mathtt{ID}; \mathtt{Streamterm}; \mathtt{Range}; \mathtt{Formula} : (\mathtt{T})\mathtt{valueterm}}$$

$(\mathtt{T} = \mathtt{T1})$

$$\frac{\begin{array}{l}(\mathtt{E}) \vdash \mathtt{Valueterm} : (\mathtt{T1})\ \mathtt{valueterm}(\mathtt{E}) \vdash \mathtt{FunctionRef} : (\mathtt{ID1})\mathtt{functionref} \\ (\mathtt{E}) \vdash \mathtt{StreamTerm} : (\mathtt{T2})\mathtt{streamterm}(\mathtt{E}) \vdash \mathtt{Range} : (\mathtt{ID2})\mathtt{range}(\mathtt{E}) \vdash \mathtt{constrseq} : \mathtt{constrseq} \\ (\mathtt{E}) \vdash \mathtt{Formula} : \mathtt{formula}(\mathtt{E}) \vdash \mathtt{ValueTerm} : (\mathtt{T3})\mathtt{valueterm}\end{array}}{\begin{array}{l}(\mathtt{E}) \vdash \mathtt{valueterm}; \mathtt{functioncall}; \mathtt{ID}; \mathtt{streamterm}; \mathtt{range}; \mathtt{constseq}; \mathtt{formula}; \mathtt{valueterm} : \\ (\mathtt{T})\mathtt{valueterm}\end{array}}$$

$\mathtt{if}(\mathtt{ID} = \mathtt{ID2})\&\&(\mathtt{T} = \mathtt{T1} = \mathtt{T2} = \mathtt{T3})$

$$\frac{(\mathtt{E}) \vdash \mathtt{Bind} : (\mathtt{E1})\mathtt{bind}(\mathtt{E} + \mathtt{E1}) \vdash \mathtt{ValueTerm} : (\mathtt{T1})\mathtt{valueterm}}{(\mathtt{E}) \vdash \mathtt{Bind}; \mathtt{ValueTerm} : (\mathtt{T})\mathtt{valueterm}}$$

$(\mathtt{T} = \mathtt{T1})$

StreamTerm

$(\mathtt{E}) \vdash \mathtt{ID} : \mathtt{streamterm}$

$$\frac{(\mathtt{E}) \vdash \mathtt{bind} : (\mathtt{E1})\mathtt{bind}(\mathtt{E} + \mathtt{E1}) \vdash \mathtt{streamterm} : (\mathtt{T1})\mathtt{streamterm}}{(\mathtt{E}) \vdash \mathtt{Bind}; \mathtt{StreamTerm} : (\mathtt{T})\mathtt{streamterm}}$$

$(\mathtt{T} = \mathtt{T1})$

$$\frac{\begin{array}{l}(\mathtt{E}) \vdash \mathtt{ValueTerm} : (\mathtt{T1})\mathtt{valueterm}(\mathtt{E}) \vdash \mathtt{FunctionRef} : (\mathtt{ID})\mathtt{functionref} \\ (\mathtt{E}) \vdash \mathtt{StreamTerm} : (\mathtt{T3})\mathtt{streamterm}(\mathtt{E}) \vdash \mathtt{Range} : (\mathtt{ID1})\mathtt{range}(\mathtt{E}) \vdash \mathtt{ConstrSeq} : \mathtt{constseq} \\ (\mathtt{E}) \vdash \mathtt{Formula} : \mathtt{formula}(\mathtt{E}) \vdash \mathtt{ValueTerm} : \mathtt{valueterm}\end{array}}{\begin{array}{l}\mathtt{ValueTerm}; \mathtt{FunctionRef}; \mathtt{ID}; \mathtt{StreamTerm}; \mathtt{Range}; \mathtt{ConstrSeq}; \mathtt{Formmula}; \mathtt{ValueTerm} : \\ (\mathtt{T})\mathtt{streamterm}\end{array}}$$

$\mathtt{if}(\mathtt{T} = \mathtt{T1} = \mathtt{T2} = \mathtt{T3})\&\&(\mathtt{ID} = \mathtt{ID1})$

$$\frac{\begin{array}{l}(\mathtt{E}) \vdash \mathtt{StreamTerm} : (\mathtt{T1})\mathtt{streamterm}(\mathtt{E}) \vdash \mathtt{Range} : (\mathtt{ID1})\mathtt{range} \\ (\mathtt{E}) \vdash \mathtt{ConstrSeq} : \mathtt{constrseq}(\mathtt{E}) \vdash \mathtt{Valueterm} : (\mathtt{T2})\mathtt{valueterm}\end{array}}{(\mathtt{E}) \vdash \mathtt{ID}; \mathtt{StreamTerm}; \mathtt{Range}; \mathtt{ConstrSeq}; \mathtt{ValueTerm} : (\mathtt{T})\mathtt{streamterm}}$$

$\mathtt{if}(\mathtt{ID} = \mathtt{ID1})\&\&(\mathtt{T} = \mathtt{T1} = \mathtt{T2})$

$$\frac{\begin{array}{l}(\mathtt{E}) \vdash \mathtt{StreamTerm} : (\mathtt{T1})\mathtt{StreamTerm}(\mathtt{E}) \vdash \mathtt{Range} : (\mathtt{ID1})\mathtt{range} \\ (\mathtt{E5}) \vdash \mathtt{ConstrSeq} : \mathtt{constrseq}(\mathtt{E}) \vdash \mathtt{ValueTerm} : (\mathtt{T2})\mathtt{valueterm}\end{array}}{(\mathtt{E}) \vdash \mathtt{ID}; \mathtt{streamterm}; \mathtt{range}; \mathtt{constrseq}; \mathtt{streamterm} : (\mathtt{T})\mathtt{valueterm}}$$

$\mathtt{if}(\mathtt{ID} = \mathtt{ID1})\&\&(\mathtt{T} = \mathtt{T1} = \mathtt{T2})$

ConstrSeq

$$(\texttt{E})\Phi \vdash \texttt{constrseq}$$

$$(\texttt{E}) \vdash \texttt{Constr} : \texttt{constr}(\texttt{E}) \vdash \texttt{ConstrSeq} : \texttt{constrseq}(\texttt{E})\texttt{Constr}; \texttt{ConstrSeq} : \texttt{constrseq}$$

Constr

$$\frac{(\texttt{E}) \vdash \texttt{Bind} : (\texttt{E1})\texttt{bind}}{(\texttt{E0}) \vdash \texttt{Bind} : \texttt{constr}}$$

$$\frac{(\texttt{E}) \vdash \texttt{Formula} : \texttt{formula}}{(\texttt{E}) \vdash \texttt{Formula} : \texttt{constr}}$$

FunctionCall

$$\frac{(\texttt{E}) \vdash \texttt{argseq} : (\texttt{T}*)\texttt{argseq}}{(\texttt{E}) \vdash \texttt{ID}; \texttt{Argseq} : (\texttt{ID})\texttt{functioncall}}$$

$$\texttt{E(ID).getTypes} = \texttt{T}*$$

Argseq

$$(\texttt{E}) \vdash \Phi : (\Phi)\texttt{argseq}$$

$$\frac{(\texttt{E}) \vdash \texttt{Arg} : (\texttt{T})\texttt{arg}(\texttt{E}) \vdash \texttt{ArgSeq} : (\texttt{T}*)\texttt{argseq}}{(\texttt{E}) \vdash \texttt{Arg}; \texttt{ArgSeq} : (\texttt{T}*)\texttt{argseq}}$$

Arg

$$(\texttt{E}) \vdash \texttt{ID} : (\texttt{T})\texttt{arg}, \texttt{E(ID).getType()}$$

FunctionRef

$$(\texttt{E}) \vdash \texttt{ID} : (\texttt{ID})\texttt{functionref}$$

5 Conclusion

This report aims to describe the type system developed for the LogicGuard specification language. In the introduction we started with describing our motivation to use a type system by showing the advantage of a type system in preventing execution errors, then we describe the separation between predicate logic specification and low level code properties which influenced the LogicGuard type system design. The rest of the report shows the detailed design aspects of the type system starting with definitions, then judgments and finally the rules derived. The language specification is attached to the report as well. We have already finished implementing the type checker based on this type system design and it has been tested and used with numerous specifications.

Appendix

Language specifications

```
ID
STRING
INT
Op = && | || | => | <=>
REL = < | <= | > | >=

Spacification = Statement*

Statement = Monitor
| Bind
| ExtTypeDec
| ExtFunctionDec
| ExtImportDec
| ExtPredicateDec
| ExtStreamDec

Monitor = ID;MonitorPositionDec*;Formula

MonitorPositionDec = ID;ID

Formula = ID
| true
| false
| ~Formula
| Formula1 Op Formula2
| Bind;Formula
| ID;Term*
| ID;StreamTerm;Range;Formula


ExtTypeDec = ID

ExtImportDec = STRING

ExtFunctionDec = Type;ID;Type*

ExtStreamDec = Type;ID

ExtPredicateDec = ID;Type*

Bind = FormulaDec;Formula
| StreamDec;StreamTerm
| PositionDec;PositionTerm
| ValueDec;ValueTerm

FormulaDec = ID;Parameter*

PositionDec = ID;Parameter*

StreamDec = ExtType;ID;Parameter*
```

```
ValueDec = ExtType;ID;Parameter*

ParameterDec = PostionParameter
| ValueParameter
| StreamParameter
| FormulaParameter

PostionParameter = ID

ValueParameter = ExtType;ID

StreamParameter = ExtType;ID

FormulaParameter = ID

Term = StreamTerm | PositionTerm | ValueTerm

StreamTerm =
ID
| ValueTerm;FunctionRef;ID;StreamTerm;Range;ConstraintSeq;Formula;Valueterm
| ID;StreamTerm;Range;ConstraintSeq;ValueTerm
| ID;StreamTerm;Range;ConstraintSeq;StreamTerm
| Bind;StreamTerm

ValueTerm =
ID
| Range
| ID;StreamTerm;Range;Formula
| ValueTerm;FunctionRef;ID;StreamTerm;Range;ConstaintSeq;Formula;ValueTerm
| Bind;ValueTerm

PositionTerm =
ID
| ID;StreamTerm;Range;Formula
| PositionTerm +/- INT
| Position
| Bind;PositionTerm

BinaryFunction = PositionTerm

Constraint = Bind | Formula

Range = PositionTerm;REL;ID
| PositionTerm;REL;ID;REL;PositionTerm

FunctionCall = ID;ParameterArgument*

Parameter = ID | FunctionCall

FunctionRef = ID
```