A New Coordination Language MediatE

Yi Li

No Institute Given

Abstract. tbd

1 Introduction

2 Overview

3 The Grammar

In this section, we mainly focus on the syntax of our language, it is divided into different parts and basically described in *Extended Backus-Naur form (EBNF)*.

Let's introduce the overview of this language first.

$$
\begin{aligned}
\langle program \rangle &::= [\, \langle statement \rangle \,]^{*} \\
\langle statement \rangle &::= \langle packageRequirement \rangle \\
&\mid \langle channel \rangle \\
&\mid \langle component \rangle \\
&\mid \langle connector \rangle
\end{aligned}
$$

3.1 Type System

The basic idea behind this type system is that we try to satisfy both researchers and programmers.

$$
\begin{aligned}
\langle primitiveType \rangle &::= \textbf{int}\, [\, \langle termint \rangle \ \textbf{..}\ \langle termint \rangle \,] \\
&\mid \textbf{float} \mid \textbf{char} \mid \textbf{bool} \\
&\mid \textbf{enum}\ \{\ \langle identifier \rangle^{+}\ \} \\
\langle type \rangle &::= \langle rawtype \rangle \\
&\mid \langle identifier \rangle \\
&\mid \textbf{array}\ \langle type \rangle\ [\ \langle termint \rangle\ ] \\
&\mid \textbf{map}\ [\ \langle type \rangle\ ]\ \langle type \rangle \\
&\mid \textbf{struct}\ \{\ (\ \langle identifier \rangle\ \textbf{:}\ \langle type \rangle\ )^{+}\ \}
\end{aligned}
$$

Basic Type. MediatEhas four built-in basic types, they are:

Int. Two types of integers are
Float.
Bool.
Enum.

Extended Type. Extended type offers an approach to contruct complex data types through the primitive ones. Three extended types are introduced as follows:

Array. An *array* T[n] is a finite ordered collection containing exactly n elements of type T.

Map. A *map* $[T_{key}]$ T_{val} is a dictionary that maps a key of type T_{key} to a value of type T_{val}

Struct. A *struct* $\{id_1 : T_1, \cdots, id_n : T_n\}$ contain n fields, each has a particular type T_i and a unique identifier id_i.

Type systems often differ greatly between formal models and real-world programming languages. For example, PRISM[1], ..

Definition 1 (Finite Types). *A type in MediatEis* finite *iff. it has a finite value domain.*

In MediatE's type system, most of basic types are finite except for the unlimited integers and doubles. Besides, extended types are also finite if they are based on finite ones.

3.2 Channels

Channels are the atomic functional units in our language. A channel compries several parameters; a set of ports, either *input* or *output*; some private variables; and a series of *ordered* transitions.

Essentially, behavior of a channel is encoded as a set of *guarded transitions.*

Before introducing the formal grammar of channels, we present a simple example here.

$$
\begin{aligned}
\langle channel \rangle &::= \textbf{channel}\ [\ \langle params \rangle\ ]\ \langle identifier \rangle\ \textbf{(}\ \langle ports \rangle\ \textbf{)} \\
&\quad\ \langle variables \rangle\ \langle transitions \rangle\ \textbf{end channel} \\
\langle ports \rangle &::= (\ \langle porttype \rangle\ \langle identifier \rangle\ )^{+} \\
\langle porttype \rangle &::= (\ \textbf{in} \mid \textbf{out}\ )\ \langle type \rangle
\end{aligned}
$$

Templates make it able to create a family of similar channels with a single definition. That is, we are able to declare a set of parameters in the channel's definition. And when creating channel instances, you have to specify concrete values to these parameters. A parameter here can be either a type identifier (decorated with prefix **type**), or a normal variable.

$$
\begin{aligned}
\langle params \rangle &::= [\ \langle param \rangle\ ^{+}] \\
\langle param \rangle &::= (\ \textbf{type} \mid \langle type \rangle\ )\ \langle identifier \rangle
\end{aligned}
$$

Variables. In the channel body, private variables can be declared in the *variable* segement and used in the *trnasition* segement. Unlike the parameters, all variables' declaration here must be followed with a initial value, which will be assigned to the variable when a MediatEsystem starts its execution.

$$\begin{aligned}\langle variables\rangle \ ::= \ & \textbf{variables} \\ & (\ \langle identifier\rangle\ \langle type\rangle\ \textbf{init}\ \langle term\rangle\ )^{+} \\ & \textbf{end variables}\end{aligned}$$

Transitions. A transition is a guarded command that is executed when

$$\begin{aligned}\langle transitions\rangle \ ::= \ & \textbf{transitions} \\ & (\ \langle transition\rangle \mid \langle group\rangle\ )^{*} \\ & \textbf{end transitions} \\ \langle transition\rangle \ ::= \ & \langle term\rangle\ \rightarrow \\ & (\ \langle statement\rangle \mid \textbf{begin}\ \langle statement\rangle^{+}\ \textbf{end}\ ) \\ \langle statement\rangle \ ::= \ & \langle identifier\rangle^{+}\ \textbf{:=}\ \langle term\rangle^{+} \\ \mid \ & \textbf{perform}\ \langle identifier\rangle^{+} \\ \langle group\rangle \ ::= \ & \textbf{group}\ \langle transition\rangle^{+}\ \textbf{end group}\end{aligned}$$

3.3 Connectors

$$\begin{aligned}\langle connector\rangle \ ::= \ & \textbf{connector}\ [\ \langle params\rangle\ ]\ \langle identifier\rangle\ (\ \langle ports\rangle\ ) \\ & [\ \langle internal\rangle\ ](\ \langle connection\rangle\ )^{+}\ \textbf{end connector} \\ \langle interal\rangle \ ::= \ & \textbf{internal}\ (\ \langle identifier\rangle\ )^{+} \\ \langle connection\rangle \ ::= \ & \langle identifier\rangle\ (\ \langle identifier\rangle^{+}\ )\end{aligned}$$

4 Semantics

5 Discussion

6 Conclusion

[2]

References

1. M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of Probabilistic Real-Time Systems," in *Proceedings of CAV 2011*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 1–6.
2. F. Arbab, "Reo: a channel-based coordination model for component composition," *Mathematical Structures in Computer Science*, vol. 14, no. 3, pp. 329–366, 2004.