Modeling Language Syntax and Semantics

The language for specifying a models consists of the following:

Defining States (simple behaviors)

A particular execution of a networked system or process can be captured as a sequence of states, where a state is a collection of attributes and their values. States can be treated as simple behaviors.

Defining Complex Behaviors

A behavior can be expressed as a sequence of one or more related states or related behaviors. A system execution is thus defined as a combination of different behaviors, and each new execution may generate a unique set of behaviors.

Defining Relationships

Language provides

1. **temporal operators** that allow expressing the ordering of states and behaviors in time without explicitly introducing time.
2. **interval-temporal operators** that allow expressing relationships like concurrency, overlap and ordering between behaviors as relationships between their time-intervals.
3. **logical operators** that allow capturing exclusivity, combination and negation relationships between behaviors.

Defining Constraints

Defining a Model

A behavior model (B_{phi}) is an assertion about the overall behavior of the system.

Defining States

States essentially capture one or more events that satisfy specified relations between attributes and their values. States can have a static attribute-value assignment or attribute values can also be dynamically determined at runtime. The state definition consists of two parts: a state identifier and a state expression. State identifiers are language variables and are used to refer to states. A state expression consists of a set of comma-separated attributes and their values.

See the complete grammar as railroad diagrams and ebnf. Below we directly give examples of state definitions.

Form	Example	Meaning	Pointer to example models, data and output
Attributes with constant values (related by equality)	state_A = {sipaddr = '10.1.11.2', dipaddr='10.1.4.2'}	Matches events whose attributes match the values specified for ***sipaddr*** and ***dipaddr***.	Example
Attributes with constant values (related by other relations)	state_B = {sport > 53}	Matches events whose ***sport*** is greater than 53	Example
Attributes with dependent values	state_C = {sipaddr=$state_A.dipaddr}	Matches events whose **sipaddr** value is same as the **dipaddr** values of state_A.	Example
Attributes with dependent and independent values	state_D = {sport=$state_B.sport, dnsqrflag=1}	Matches events whose **sport** value is same as the **sport** value of state_B and whose ***dnsqrflag*** has constant value 1.	Example
Importing states from other models	state_E = DNSREQRES.dns_req()	***dns_req()*** state is directly imported from the existing DNSREQRES model.	Example
Importing behaviors from other models	state_F = DNSREQRES.DNS_REQ_RES()	The entire ***DNS_REQ_RES()*** model is directly imported from the existing DNSREQRES model.	Example
Importing behaviors from other models with customization of attributes	state_G = DNSREQRES.DNS_REQ_RES(sport > 30000)	The entire ***DNS_REQ_RES()*** model is directly imported from the existing DNSREQRES model and the ***sport*** attribute is customized.	Example

Wildcarding of string values	state_H = {sipaddr='10.1.*'}	This state matches events which have ***sipaddr*** starting with ***10.1.***	Example

Important points

- The attributes specified for the states should match the attribute names that are valid for each eventtype.
- Constant values like strings are required to be single quoted.
- The attributes and values can be related by the following six relations: **('=', '!=', '>', '>=', '<', '<=')**.
- By default, every state definition will match 1 event by default.

Defining Behaviors

Expressing a behavior in the language constitutes writing sub-formulae in the modeling logic. Behaviors are always enclosed within parenthesis ?(? and ?)?. Simple behaviors are constructed by relating one-or-more state propositions using operators, while complex behaviors are constructed by relating one-or-more behaviors.

Behaviors are evaluated left to right and operator precedence is defined explicitly using the grouping operators.

Defining Relationships

Let **A** and **B** represent either states or behaviors.

Examples to be updated shortly

Operator symbol	**Meaning**	**Example**	**Pointer to example models, data and output**
A **~>** B	B occurs after A , that is, whenever A is satisfied B will eventually be satisfied.		
A **olap** B	A overlaps B		
A **sw** B	A starts with B		
A **ew** B	A ends with B.		
A **eq** B	A is equal to B in duration.		
A **dur** B	A is true during B.		
A **and** B	Both A and B are true.		
A **or** B	A or B are not both false simultaneously.		
A **xor** B	Either A or B is true but not both.		
not A	A is not true.		

Defining Constraints

This section will be updated shortly.

Defining a Model

This section will be updated shortly
