

NAME

ASTG – Asynchronous Signal Transition Graphs

SYNOPSIS

An Asynchronous Signal Transition Graphs Interchange Format with support for underlying bounded Petri nets and MIX-MAX delay model.

DESCRIPTION

The ASTG format derived from the classical Signal Transition Graph model originally development in *SIS*, but extended with abstract labeling.

A STG is a bipartite directed graph that contains an input/output/internal signal declaration, active/passive/internal label declaration and a set of transitions and places implemented as an adjacency list to describe the net interconnection structure.

Each transition is associated to a signal or label. Signal transitions are additionally associated to positive or negative switching of the signal.

All names in the ASTG description must start with a letter, consist of letters, digits and underscores, and are case-sensitive. A signal transition is represented with a suffix: "+" means a low to high transition, "-" means high to low, "" means toggles (changes to the opposite value). Label transitions do not require any suffix.

The ASTG format contains the following keywords:

.model <model-name>

This gives an arbitrary name to the ASTG, and it must be the first line of the model description.

.inputs <signal-list>

Specifies a list of names of PN input signals. Signals from multiple *.inputs* are concatenated.

.outputs <signal-list>

Specifies a list of names of PN output signals. Signals from multiple *.outputs* are concatenated.

.internal <signal-list>

Specifies a list of names of PN internal signals, i.e. signals which are only used to maintain state information.

.dummy <name-list>

Specifies a list of names which are accepted as dummy or null transitions. Null transitions are necessary to specify some behaviors using the ASTG syntax. By convention, the name "e" is used as a dummy signal to represent epsilon transitions. (Used for backward compatibility).

.input_labels <label-list>

Specifies a list of names of ASTG input labels (not associated to any digital signal). Labels from multiple *.input_labels* are concatenated.

.output_labels <label-list>

Specifies a list of names of ASTG output labels (not associated to any digital signal). Labels from multiple *.output_labels* are concatenated.

.internal_labels <label-list>

Specifies a list of names of ASTG internal labels (not associated to any digital signal). Labels from multiple *.internal_labels* are concatenated.

.input_places <place-list>

Specifies a list of names of ASTG input places (visible in the ASTG interface). Places from multiple *.input_places* are concatenated.

.output_places <place-list>

Specifies a list of names of ASTG output places (visible in the ASTG interface). Places from multiple *.output_places* are concatenated.

.graph

Indicates the lines which follow describe the ASTG net structure using an adjacency list format. This must follow all signal and label declarations (*.inputs*, etc.). ASTG places are optional for simple constraints between two transitions; in this case an intervening place is generated automatically. Multiple instances of a transition are distinguished by following them with a slash and a copy number. For example, a second instance of signal transition "t+" can be specified by "t+/2". The third instance of label transition "req" can be specified by "req/3". Copy numbers do not have to be consecutive.

Weighted and inhibitor arcs:

```
.graph
a+ p1(2) b+ p2
p2 d-(0)
```

indicates that the event *a+* has arcs to

- place *p1* with weight 2
- transition *b+* with weight 1 (implicit places cannot have weighted arcs)
- place *p2* with weight 1 (default)

and that place *p2* has an inhibitor arc (weight 0) to transition *d-*

.marking {<place & num_tokens list>}

An initial marking can optionally be specified after the net structure has been given. Implied places (see *.graph*) between two transitions *x* and *y* can be specified using the syntax <*x,y*>. Bounded initial markings are specified by indicating {<place>=num_tokens} (default is 1).

```
.marking {p1 <a+,b+> p2=2 <c+,d->=3}
```

indicates that *p2* has 2 tokens and <*c+,d-*> 3 tokens, while *p1* and <*a+,b+*> have one token.

.capacity {<place & max_capacity list>} (optional)

Indicates the maximum capacity for each place (default is 1). This is intended for overflow detection during token flow analysis.

```
.capacity p2=3 <c+,d+>=4
```

indicates that *p2* has a maximum capacity for three tokens, and <*c+,d+*> has a maximum capacity for four tokens.

.initial_state {<signal-list>} (optional)

When the underlying Petri net is interpreted as an STG, this option defines the values of the signals in the initial marking of the net. This option is useful when toggle transitions are used and the initial value of some signal cannot be deduced from the flow analysis of the ASTG. When necessary, it is assumed the value 0 for signals with undefined initial value. In the example, the initial values

```
.initial_state !a b
```

indicates that the initial values for signals *a* and *b* are defined to be 0 and 1 respectively.

.delay {<transition-delay-list>} (optional)

This option defines the values of the delays associated to the transitions in the STG. We assume a min-max delay model. Transitions without assigned delay assume a zero to infinity delay range.

.delay R+ = (10,20) G+ = 15

indicates that transition *R+* has a delay range between 10 and 20 delay units, while *G+* has 15 delay units, i.e. a delay range between 15 and 15.

.end

This required line indicates the end of the *ASTG* description.

Error messages are printed for any unrecognized input sequences.

FILES

SEE ALSO

The **TranSyT** WEB page at: <http://research.ac.upc.es/VLSI/transyt/transyt.html>

The **Petrify** WEB page at: <http://www.lsi.upc.es/~jordicf/petrify/>

hcastg(1).

BIBLIOGRAPHY

Enric Pastor, Oriol Roig, Jordi Cortadella, and Rosa M.Badia, *Petri net Analysis Using Boolean Manipulation*, 15th International Conference on Application and Theory of Petri Nets, pages 416-435, June 1994.

Oriol Roig, Jordi Cortadella, and Enric Pastor, *Verification of asynchronous circuits by BDD-based model checking of Petri nets*, 16th International Conference on Application and Theory of Petri Nets, pages 374-391, June 1995.

Alex Kondratyev, Jordi Cortadella, Michael Kishinevsky, Enric Pastor, Oriol Roig and Alex Yakovlev, *Checking Signal Transition Graph implementability by symbolic BDD traversal*, European Design and Test Conference (EDAC-ETC-EuroASIC), pages 325-332, March 1995.

BUGS

AUTHOR

Enric Pastor
Department of Computer Architecture
Universitat Politecnica de Catalunya
Barcelona, Spain
enric@ac.upc.es