Lazy Data Saturation

Jasper Berendsen

Radboud University Nijmegen

November 20, 2007

Example: Very simple model of Zeroconf protocol

- Protocol for dynamic configuring IP addresses
- A large number of hosts communicate via broadcasts
- Property: *Hosts should not use the same IP address*
- Main simplifications
  - No probabilities
  - No delay in the network

(on my website paper on Zeroconf)

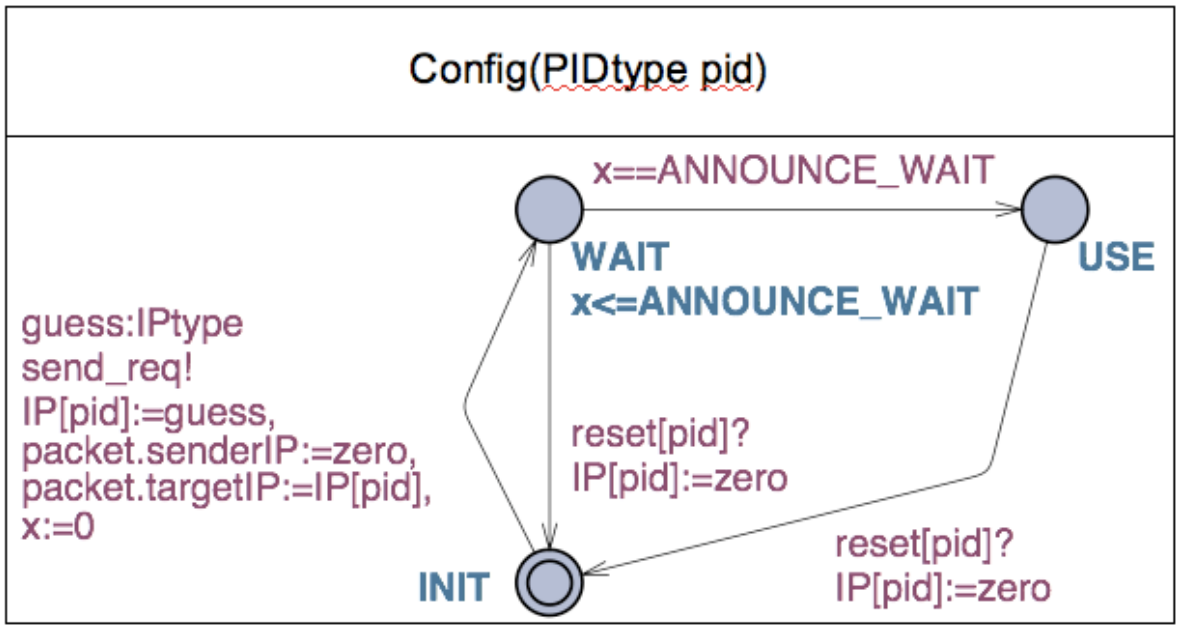
Model of a single host

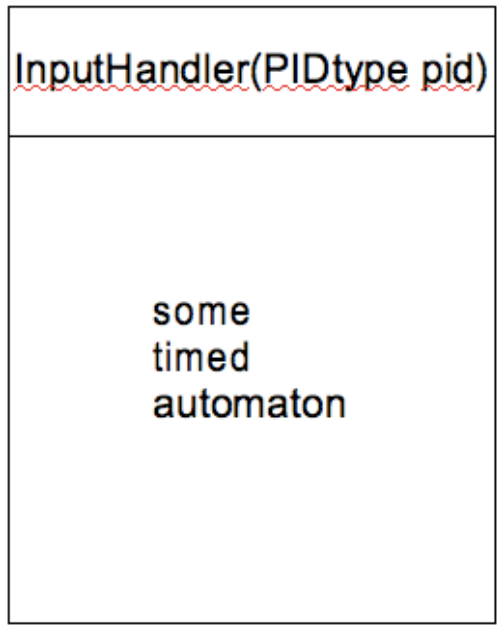
Globals

```
typedef scalar[k] PIDtype;
typedef scalar[m+1] Iptype;

typedef struct{     IPtype senderIP; // sender IP address
                    IPtype targetIP; // target IP address
} ARP_packet;

IPtype zero;
ARP_packet packet;
```





Restrictions on use of scalarsets [ID93]

- The elements can only be compared on equality, no addition or other operators
- Every scalarset is its own type incomparable to other scalarset types
- Iterating over a scalarset variable is order independent

Symmetry Reduction on scalar PIDtype

For explanation: no InputHandler automata, and no clocks
For example $k = 3$ hosts
State vectors:

$$\begin{array}{r} \text{Config}(h_1).\text{loc} \\ \text{Config}(h_1).\text{guess} \\ \text{Config}(h_2).\text{loc} \\ \text{Config}(h_2).\text{guess} \\ \text{Config}(h_3).\text{loc} \\ \text{Config}(h_3).\text{guess} \\ \text{packet.senderIP} \\ \text{packet.targetIP} \end{array} \begin{pmatrix} \text{INIT} \\ \text{zero} \\ \text{WAIT} \\ \text{ip}_1 \\ \text{USE} \\ \text{ip}_2 \\ \text{zero} \\ \text{ip}_1 \end{pmatrix} \approx \begin{pmatrix} \text{INIT} \\ \text{zero} \\ \text{USE} \\ \text{ip}_2 \\ \text{WAIT} \\ \text{ip}_1 \\ \text{zero} \\ \text{ip}_1 \end{pmatrix}$$

Symmetry Reduction on scalar IPtype

State vectors:

$$\begin{array}{r} \text{Config}(h_1).\text{loc} \\ \text{Config}(h_1).\text{guess} \\ \text{Config}(h_2).\text{loc} \\ \text{Config}(h_2).\text{guess} \\ \text{Config}(h_3).\text{loc} \\ \text{Config}(h_3).\text{guess} \\ \text{packet.senderIP} \\ \text{packet.targetIP} \end{array} \begin{pmatrix} \text{INIT} \\ \text{zero} \\ \text{WAIT} \\ \text{ip}_1 \\ \text{USE} \\ \text{ip}_2 \\ \text{zero} \\ \text{ip}_1 \end{pmatrix} \approx \begin{pmatrix} \text{INIT} \\ \text{zero} \\ \text{WAIT} \\ \text{ip}_2 \\ \text{USE} \\ \text{ip}_1 \\ \text{zero} \\ \text{ip}_2 \end{pmatrix}$$

Data saturation [ID93]

Suppose the number of hosts is fixed to 3.

- Verification makes no difference for more that 6 IP addresses available.
  Intuition: Simply look at the state vector.
- But behaviour shows that at most 3 IP addresses are needed.
  Intuition: packet.senderIP and packet.targetIP are only set to IPs already in use.

IPtype is a so called data scalar.

Data scalarsets

Extra conditions:

- not used as array indices
- not used for template arguments
- not used in for-statements

Check all instances?

In the example: data saturation for > 6#IPs
All instances of the model ($\#\text{IP} = 1, = 2, \ldots, = 6$) must be verified.

Example

Idea: use only guards having only existential quantification in their Negation Normal Form

Lazy Data Saturation

In *selection* and *quantification* over data scalars, elements of the scalar are choosen.

Idea: either

1. pick a value that is not used anywhere in the state vector
2. or pick some used value

State vectors:

$$
\begin{array}{r}
\mathrm{Config}(h_1).\mathrm{loc} \\
\mathrm{Config}(h_1).\mathrm{guess} \\
\mathrm{Config}(h_2).\mathrm{loc} \\
\mathrm{Config}(h_2).\mathrm{guess} \\
\mathrm{Config}(h_3).\mathrm{loc} \\
\mathrm{Config}(h_3).\mathrm{guess} \\
\mathrm{packet.senderIP} \\
\mathrm{packet.targetIP}
\end{array}
\begin{pmatrix}
\mathrm{INIT} \\ \mathrm{zero} \\ \mathrm{WAIT} \\ \mathrm{ip}_1 \\ \mathrm{USE} \\ \mathrm{ip}_2 \\ \mathrm{zero} \\ \mathrm{ip}_2
\end{pmatrix}
\longrightarrow
\begin{pmatrix}
\mathrm{WAIT} \\ ? \\ \mathrm{WAIT} \\ \mathrm{ip}_1 \\ \mathrm{USE} \\ \mathrm{ip}_2 \\ \mathrm{zero} \\ ?
\end{pmatrix}
$$

Future Work

- Implement for Uppaal model checker
- Investigate if relaxation of *Diagonal property* is possible.


C.N. Ip and D.L. Dill.
Better verification through symmetry.
In David Agnew, Luc J. M. Claesen, and Raul Camposano, editors, *Proceedings of the 11th IFIP WG10.2 International Conference on Computer Hardware Description Languages and their Applications - CHDL '93,* Ottawa, Ontario, Canada, 26-28 April, 1993, volume A-32 of *IFIP Transactions*, pages 97–111. North-Holland, 1993.