T-79.186 Reactive Systems

Spring 2003, Lecture 8

Keijo Heljanko

February 25, 2003

The algorithm creates a graph stored in the set "$nodes$". The nodes (states) of this graph are labelled with formulas. Actually, from now on we are only interested in the formulas stored in the set "Old".

It is now easy to obtain a Büchi automaton from this graph. (Using a slightly different Büchi automata definition than what has been used in this course so far.)

First of all a special initial state "$init$" is created. This state is the only state in S^0.

All nodes "$p \in$ nodes" together with the initial state "$init$" are the states S of the Büchi automaton.

The labelling of the arcs of the Büchi automaton can be derived from the formula labelling of states.

Namely, a state is compatible with a set of valuations as described below.

A valuation $v \in 2^{AP}$ is compatible with the label of a node s iff:

- $\forall p \in AP$: if $p \in$ s.Old then $p \in v$, and
- $\forall p \in AP$: if $\neg p \in$ s.Old then $p \notin v$.

There is an arc from a state $s \in S$ to a state $r \in S$ with a label $v \in 2^{AP}$ iff

- v is compatible with the valuation of r, and
- $s \in r$.Incoming.

The Büchi automaton class used is called **generalized Büchi automata**. In this class the acceptance component consist of several acceptance sets. The basic idea is that an accepting run should visit some accepting state from each acceptance set infinitely often.

More formally the acceptance component $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$, where each $F_i \subseteq S$.

Now a generalized Büchi automaton $\mathcal{A}$ accepts a run r iff for all $F_i \in \mathcal{F}$:
$\mathit{inf}(r) \cap F_i \neq \emptyset$.

Note that in the special case $\mathcal{F} = \emptyset$ all infinite runs of the generalized Büchi automaton are accepting.

We will to rule out the use of case b) infinitely many times without also using case a) infinitely many times when proving the until formula $f_1 \, U \, f_2 \in sub(f)$. This is done by using one acceptance set for each until formula.

Assume, that the (until) subformulas (subformulas of the form $f_1 \, U \, f_2$) are numbered $1, 2, \ldots, n$. Then the acceptance component $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$, where:

For each $1 \leq i \leq n$ the state s belongs to F_i iff

- $right(f_i) \in s$.Old, or
- $f_i \notin s$.Old.

These together will assure that for each until formula f_i either the right hand side is eventually proved, or that we do not have the until formula f_i as our proof obligation from state s onwards.

Note also that if there are no until formulas, $\mathcal{F} = \emptyset$.

Many of the emptiness checking algorithms, for example the nested depth first search, do not handle generalized Büchi automata. Thus most of the LTL to Büchi translation algorithms make a (non-generalized) Büchi automaton $\mathcal{A}'$ out of the generalized Büchi automaton with the following procedure (which works for any generalized Büchi automaton).

Another option is to use e.g., a MSCC based emptiness checking algorithm instead.

Definition 8.7 Let $\mathcal{A}$ be a generalized Büchi automaton $(\Sigma, S, S^0, \rho, \mathcal{F})$, where $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$. We now define a (non-generalized) Büchi automaton $\mathcal{A}'$ based on the number of sets in $\mathcal{F}$ as follows:

- $\mathcal{F} = \emptyset$: $\mathcal{A}' = (\Sigma, S, S^0, \rho, S)$,
- $\mathcal{F} = \{F_1\}$: $\mathcal{A}' = (\Sigma, S, S^0, \rho, F_1)$,
- $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$, where $n \geq 2$: $\mathcal{A}' = (\Sigma, S', {S^0}', \rho', {F_1}')$, where:
  - $S' = S \times \{1, 2, \ldots, n\}$,
  - ${S^0}' = S^0 \times \{1\}$,
  - ρ' is defined as follows: $(s', j) \in \rho'((s, i), a)$ iff $s' \in \rho(s, a)$ and $((s \notin F_i \text{ and } j = i) \vee (s \in F_i \text{ and } j = (i \, mod \, n) + 1))$, and
  - ${F_1}' = F_1 \times \{1\}$.

Now it holds that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$, and $\mathcal{A}'$ is never smaller than $\mathcal{A}$. In fact, in the worst case $\mathcal{A}'$ has n times as many states as $\mathcal{A}$, where n in the number of acceptance sets. (This construction is essentially optimal, result due to H. Tauriainen.)

Now we can prove some worst-case bounds on the automata size based on the presented translation algorithm.

Given an LTL formula f in negation normal form, for the generalized Büchi automaton $\mathcal{A}_f$ a (coarse) upper bound on the number of states it has is $1 + 2^{(2 \cdot |sub(f)|)}$ states.

(There is the state "$init$", plus at most as many different states as there are possible combinations of "Old" and "Next" sets, of which there are at most $2^{(2 \cdot |sub(f)|)}$.)

Theorem 8.8 Given an LTL formula f in negation normal form, the generalized Büchi automaton $\mathcal{A}_f$ has at most $2^{\mathcal{O}(|f|)}$ states.

Converting the generalized Büchi automaton to a non-generalized one we get (a coarse) upper bound of $|sub(f)| \cdot (1 + 2^{(2 \cdot |sub(f)|)})$ states. We thus get also the following result.

Theorem 8.9 Given an LTL formula f in negation normal form, the (non-generalized) Büchi automaton $\mathcal{A}_f$ has at most $2^{\mathcal{O}(|f|)}$ states.

There are quite a few highly optimized freely available LTL to Büchi automata translators available. Rolling your own can be educational, but not likely very effective in terms of the final Büchi automaton size.

The exponential blow-up in the construction is unavoidable when translating into Büchi automata. For example model checking the LTL formula

$$(strong\_fairness) \rightarrow (property)$$

will exhibit this behavior, when one starts growing the value of n in

$$strong\_fairness = \bigwedge_{1 \leq i \leq n} (\Box\Diamond p_i \rightarrow \Box\Diamond q_i).$$

Such fairness assumptions sometimes arise in model checking of liveness properties.