Reducing Acceptance Marks in Emerson-Lei Automata

Tereza Schwarzová and Jan Strejček

Masaryk University, Brno, Czech Republic
{xschwar3,strejcek}@mail.muni.cz

Abstract. The formalism called *transition-based Emerson-Lei automata (TELA)* generalizes many traditional kinds of automata over infinite words including Büchi, co-Büchi, Rabin, Streett, and parity automata. Transitions in TELA are marked with acceptance marks and accepting formulas are boolean combinations of terms saying that a particular mark has to be visited infinitely or finitely often in an accepting run. Algorithms processing these automata can be very sensitive to the number of different acceptance marks. We introduce two techniques reducing the number of acceptance marks in a TELA without altering its structure and the represented language. One technique represents a standard approach based on relations among acceptance marks in individual strongly connected components of the automaton. The other technique constructs quantified boolean formula (QBF) queries that ask a QBF solver for an acceptance condition with fewer acceptance marks and the placement of these marks in the automaton structure. Both techniques are implemented and experiments show that the number of acceptance marks in TELA produced by state-of-the-art tools ltl3tela, Rabinizer 4, and SPOT can be sometimes reduced. In the case of Rabinizer 4, we reduced the cummulative number of acceptance marks to less than one half.

1 Introduction

- Emerson-Lei automata jsou v principu staré, ale v poslední době znovuobjevene [?] a je o tom ted spousta clanku [?] a nastroju
- algoritmy jsou obvykle citlive na akceptacni podminku (citace)
- cilem prace je zjednodusit akceptacni podminku (zejmnena redukovat pocet akceptacnich znacek) bez zmeny struktury automatu

2 Preliminaries

2.1 TELA

A *transition-based Emerson-Lei automaton* over alphabet Σ is a tuple $\mathcal{A} = (\mathcal{Q}, M, \Sigma, q_0, \delta, \varphi)$, where

- $\mathcal{Q}$ is a finite set of *states*,

- M is a finite set of *acceptance marks*,
- Σ is a finite *alphabet*,
- q_0 is an *initial state*,
- $\delta \subseteq \mathcal{Q} \times \Sigma \times 2^M \times \mathcal{Q}$ is a transition relation,
- φ is the acceptance condition constructed by the following abstract syntax equation, where $m \in M$:

$$\varphi ::= \mathit{true} \mid \mathit{false} \mid \mathsf{Inf}(m) \mid \mathsf{Fin}(m) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi).$$

A run π of $\mathcal{A}$ on word $u = u_0u_1u_2 \cdots \in \Sigma^\omega$ is an infinite sequence of transitions $\pi = (q_0, u_0, M_0, q_1)(q_2, u_1, M_1, q_2) \cdots \in \delta^\omega$.

Let $M(\pi)$ denote set of all acceptance marks, that occur in run π in infinitely many transitions. The run satisfies $\mathsf{Inf}(m)$ *iff* $m \in M(\pi)$ and it satisfies $\mathsf{Fin}(m)$ *iff* $m \notin M(\pi)$. Run π is *accepting* if it satisfies the formula φ. A set of words, which are accepted by $\mathcal{A}$, represent a language $\mathcal{L}(\mathcal{A})$.

Sometimes the set of transitions with the same acceptance mark $m \in M$ is referred to as *acceptance set* $Z(m)$.

- define DNF

2.2 Strongly connected component

Let us consider a TELA $\mathcal{A} = (\mathcal{Q}, M, \Sigma, q_0, \delta, \varphi)$. *Strongly connected component* (SCC) of $\mathcal{A}$ is each maximal set of states $S \subseteq Q$ such that any two states $q_1, q_2 \in S$ are connected by a sequence of transitions. Let us enumerate the SCCs of $\mathcal{A}$. Set of all transitions of the SCC S_i is $\delta_{S_i} = \delta \cap (S_i \times \Sigma \times 2^M \times S_i)$. A run of SCC S_i is a run of automaton $\mathcal{A}$ with the infinite suffix contained within S_i.

2.3 Quantified Boolean formula

Quantified Boolean formulae (QBFs) is an extension of propositional logic – some Boolean variables in the formula can be quantified with *universal* or *existential* quantifiers. Formula Φ is a QBF formula in *prenex normal form* $\iff$ $\Phi = Q_1x_1...Q_nx_n.\varphi$, where $Q_i \in \{\forall, \exists\}$, x_i is a quantified Boolean variable and φ is quantifier-free Boolean formula.

Let Φ be a QBF. Semantically we define the *universal* quantifier as $\forall x.\Phi = \Phi[x \to \mathit{true}] \wedge \Phi[x \to \mathit{false}]$ and the *existential* quantifier as $\exists x.\Phi = \Phi[x \to \mathit{true}] \vee \Phi[x \to \mathit{false}]$.

Let X be a finite set of variables. Assignment $\mathcal{A}_X : X \to \{\mathit{false}, \mathit{true}\}$ is a total mapping of variables defined on set X to Boolean values. Let Φ be a QBF. If there is an assignment $\mathcal{A}_X$ that evaluates formula Φ as *true*, then Φ is *satisfiable* (SAT). A set $S \subseteq X$ is a *model* of Φ if the assignment $\mathcal{A}_X$ maps *true* to every element of S. If there is no such assignment that evaluates the formula as *true*, we say the formula is *unsatisfiable* (UNSAT).

3 Reduction based on acceptance sets relations / SCC-based reduction

In this section, the simplification strategy is to find relations between acceptance sets and reduce the redundant ones. SCC-based simplification is the coarsest simplification we propose.

Let $\mathcal{A}$ be an original automaton and let ψ be an acceptance condition of $\mathcal{A}$. Every run of $\mathcal{A}$ has an infinite suffix that takes place within one SCC S_i. Thus the evaluation of ψ depends purely on the SCC S_i and we can optimize the acceptance condition for each SCC separately. :w This optimization consists of removing redundant terms from acceptance condition and relabeling of acceptance marks on the transitions. The state and transition structure of the SCC does not change. This way we obtain set of simplified acceptance conditions that we merge into new acceptance formula ψ'. The automaton with relabeled transitons and new acceptance condition ψ' we denote $\mathcal{A}'$. Finally, we ensure that automaton $\mathcal{A}'$ is equivalent to the original automaton $\mathcal{A}$.

3.1 S_i simplification

At first, we remove the unused acceptance marks and transform the SCC's acceptance condition ψ_i to *disjunctive normal form* (DNF), so we have a unified shape of the condition. If TELA acceptance condition is in DNF, two distinct terms in one disjunct can only appear in three possible forms:

- $\mathsf{Inf}\,i \land \mathsf{Inf}\,j$
- $\mathsf{Fin}\,i \land \mathsf{Fin}\,j$
- $\mathsf{Fin}\,i \land \mathsf{Inf}\,j$

Since formula ψ_i is in DNF, we can represent it as a set of disjuncts $\overline{\psi_i} = \{D_1, D_2, \dots D_k\}$.

Let $D_k \in \overline{\psi_i}$ be a disjunct of formula $\overline{\psi_i}$ and $C_j \in D_k$ a conjunct of disjunct D_k. Furthermore let $i, j, k, l \in M$ be distinct acceptance marks that occur in $\mathcal{A}$.

In the next section, some properties of an automaton allow us to substitute a Boolean value *true* or *false* for a particular term of the formula ψ_i. The consequences of this substitution are divided into a number of cases. We represent it on the set-format of formula we just defined.

- If C_j is substituted by *true*, the conjunct is omitted from D_k. Thus $D_k = D_k \setminus \{C_j\}$.
- If C_j is substituted by *false*, the conjunct causes that the whole D_k evaluates to *false*. Thus $\overline{\psi_i} = \overline{\psi_i} \setminus \{D_k\}$.

That being said, we can define the reduction techniques.

Acceptance formula modifications

Since we optimize for each SCC separately, the acceptance formula corresponding to SCC S_i can contain terms with acceptance marks that are not present on the transitions of S_i. Let $k \in M$ be an acceptance mark that is not present on any edge of S_i. Then any term that contains this acceptance mark in $\overline{\psi_i}$ can be immediately substituted with a boolean value and thus removed from $\overline{\psi_i}$.

Therefore acceptance mark k is not visited in any run of S_i and thus every term in form $\mathsf{Fin}\,k$ is substituted with *true* and every term $\mathsf{Inf}\,k$ is substituted with *false*.

On the contrary, let j be an acceptance mark that is present on every transition of S_i. That means that at least one transition with j is visited infinitely often by a run of $\mathcal{S}_{\rangle}$. Thus every term $\mathsf{Fin}\,j$ is substituted by *false* and every term in form $\mathsf{Inf}\,j$ is substituted by *true*.

Inf conjuncts

Reduction of a conjunct of $\mathsf{Inf}\,k$ form is based on the inclusion of the sets of edges labeled with k and set of edges labeled with j. If all transitions that contain an acceptance mark k also contain j, we can remove $\mathsf{Inf}\,j$ from all disjuncts, where it occurs together with $\mathsf{Inf}\,k$. More formally, if the following conditions hold:

- $Z(k) \subseteq Z(j)$
- $\mathsf{Inf}\,k, \mathsf{Inf}\,j \in D_k$

then we can we can substitute $\mathsf{Inf}\,j$ with Boolean value *true*. This modification does not change the language because it does not affect accepting runs of S_i. The transitions can create an accepting run that has the potential to satisfy D_k only if they are labeled with both k and j. Since $Z(k) \subseteq Z(j)$, one can notice that that if a run satisfies $\mathsf{Inf}\,k$ then $\mathsf{Inf}\,j$ is satisfied as well. Therefore if we remove $\mathsf{Inf}\,j$ from $\overline{\psi_i}$, this modification does not affect the language.

Fin conjuncts

Similarly as in previous section, this reduction is based on inclusion. If following conditions hold:

- $Z(k) \subseteq Z(j)$
- $\mathsf{Fin}\,k, \mathsf{Fin}\,j \in D_k$

then we can we can substitute $\mathsf{Fin}\,k$ with boolean value *true*. This modification does not change the language because from if j is visited finitely often, then also k is visited finitely often.

We can merge two conjuncts into one if the conjuncts always occur in the same disjuncts in Fin form. We can do so regardless of the position of the acceptance marks that are contained in the conjuncts. If the following condition is met:

– $\forall D_k : \mathsf{Fin}ⓚ \in D_k \iff \mathsf{Fin}ⓙ \in D_k$

We introduce a fresh acceptance mark ⓛ, such that $Z(ⓛ) = Z(ⓚ) \cup Z(ⓙ)$. Finally, we can update D_k:

$$D_k = (D_k \setminus \{\mathsf{Fin}ⓚ, \mathsf{Fin}ⓙ\}) \cup \{\mathsf{Fin}ⓛ\}$$

This update basically means that $\mathsf{Fin}ⓚ, \mathsf{Fin}ⓙ$ are substituted by *true* and new conjunct $\mathsf{Fin}ⓛ$ is added to D_k. The acceptance mark ⓙ is placed on every edge of the SCC S_i that is labeled with ⓚ or ⓙ. This modification does not change the language because it does not change the acceptance runs of S_i. It only relabels the acceptance marks on edges that can be seen finitely often.

Fin ∧ Inf conjuncts
There are two cases when we can reduce one of the pair of $\mathsf{Inf}ⓘ$ and $\mathsf{Fin}ⓛ$ conjuncts.

At first, if every transition labeled with ⓘ is also labeled with ⓛ, the conjunction of $\mathsf{Inf}ⓘ \wedge \mathsf{Fin}ⓛ$ is never *true* because every run that visits ⓛ also visits ⓘ. Therefore we can reduce the whole disjunct D_k. If the following conditions are met:

– $Z(ⓘ) \subseteq Z(ⓛ)$
– $\mathsf{Inf}ⓘ, \mathsf{Fin}ⓛ \in D_k$

We substitute the whole disjunct D_k for *false*. In other words $\overline{\psi_i} = \overline{\psi_i} \setminus \{D_K\}$.

Finally we can simplify $\overline{\psi_i}$, if the edges labeled with ⓘ and edges labeled with ⓛ are complementary. More formally if:

– $\mathsf{Inf}ⓘ$ and $\mathsf{Fin}ⓛ \in D_k$
– $Z(ⓘ) \cap Z(ⓛ) = \emptyset \wedge Z(ⓘ) \cup Z(ⓛ) = \delta_{S_i}$

Then we can substitute *true* for $\mathsf{Inf}ⓘ$. This reduction does not change the language because if during a run of S_i all transitions labeled with ⓛ are visited only finitely often, then necessarily at least one transition labeled with ⓘ is visited infinitely often. Therefore removing $\mathsf{Inf}ⓘ$ from $\overline{\psi_i}$ does not affect the language, since the validity of $\mathsf{Fin}ⓛ$ implies the validity of $\mathsf{Inf}ⓘ$.

3.2 Merge of ψ_i

After applying reduction techniques from the previous section, each S_i has its own acceptance condition ψ_i. The goal of this section is to merge these acceptance conditions into one formula ψ' with emphasis on obtaining a formula with the minimal number of acceptance sets.

At first, the algorithm finds ψ_i with the greatest number of disjuncts and states it as a base of the new acceptance condition ψ'. Now the algorithm continues by the successive merging of ψ' with the unmerged formula ψ_i that has the greatest number of disjuncts. This repeats until all formulae ψ_i are merged. This process updates the form of ψ' until it reaches its final form. The process of merging two formulae consists of two phases. In the first phase a suitable pairing of disjuncts is found and in the second phase, these disjuncts are merged.

1. Let ψ' denote the future acceptance condition of $\mathcal{A}'$ and ψ_i is the acceptance condition of S_i chosen to be merged. Then the algorithm uses *linear sum assignment* to determine which disjunct of ψ_i is merged with which disjunct of ψ'. The process of using *linear sum assignment* and the pairing procedure is described in detail in work by T. Šťastná [?].
2. Let $D_K \in \overline{\psi_i}$ be a disjunct paired with $D_L \in \overline{\psi'}$. All conjuncts of D_L are initially labeled as unused. Now the algorithm maps every conjunct $C_k \in D_K$ to a suitable conjunct $C_l \in D_L$ and labels C_l as used. [1] A conjunct C_l is suitable for C_k is it is on the same type (Fin or Inf), it is labeled as unused. If no suitable C_l is found, the algorithm adds C_k to disjunct D_L and marks it as used.

In an acceptance mark $m_l \in M$ occurs in ψ' in more than one conjunct, we need to check whether all of these conjuncts are mapped to conjuncts form ψ_i with the same acceptance mark $m_k \in M$. If not, we resolve this conflict by replacing all additional occurrences of m_l with fresh acceptance mark m_n and we place m_n on the exact transition of $\mathcal{A}'$ where m_l is placed. Finally, in the end, the algorithm removes every acceptance mark that is not present in ψ' from the edge of $\mathcal{A}$.

Example 1. Consider $\overline{\psi'} = \{\{\mathsf{Fin}\textcircled{0}, \mathsf{Inf}\textcircled{1}\}, \{\mathsf{Fin}\textcircled{0}\}\}$ and $\overline{\psi_i} = \{\{\mathsf{Fin}\textcircled{4}\}, \{\mathsf{Fin}\textcircled{2}, \mathsf{Inf}\textcircled{3}\}\}$. We obtain the pairing $(1,2),(2,1)$, meaning that the first disjunct of $\overline{\psi_i}$ is merged wit the second disjunct of $\overline{\psi'}$ and second disjunct of $\overline{\psi_i}$ is merged with the first disjunct of $\overline{\psi'}$. Then $\mathsf{Fin}\textcircled{0}$ is mapped to $\mathsf{Fin}\textcircled{2}$, $\mathsf{Inf}\textcircled{1}$ is mapped to $\mathsf{Inf}\textcircled{3}$ and $\mathsf{Fin}\textcircled{0}$ is again mapped to $\mathsf{Fin}\textcircled{4}$ which causes a conflict. Therefore we need to replace the second occurrence of $\textcircled{0}$ with fresh mark $\textcircled{5}$. The set-representation of formula ψ' after merging is $\psi' = \{\{\mathsf{Fin}\textcircled{0}, \mathsf{Inf}\textcircled{1}\}, \{\mathsf{Fin}\textcircled{5}\}\}$.

Unfortunately, sometimes when the optimal mapping was not found, the procedure produces an automaton with a more complex acceptance condition than the acceptance condition of the input automaton. Therefore we propose and implement an optimization, which prevents such a behavior and in general, produces smaller acceptance condition than the original one.

- The selection of base of ψ' is based on two keys. The primary key is cardinality (number of clauses) of $\overline{\psi_i}$, the secondary key is cardinality of clauses (number of terms in clause).
- We order the disjunct in every $\overline{\psi_i}$ in descending order, where the key is the cardinality of a particular disjunct.
- The DNF conversion of the input acceptance condition can lead to an exponential blowup of the formula, where some acceptance marks occur more than once. Since the reduction of acceptance sets is basically denoting the redundant ones as *true* or *false*, we can as well easily obtain simplified acceptance formula in CNF (we simply denote the same acceptance sets as *true*

[1] The indexes differ to emphasize the fact that they are not equal.

or *false* as we did in DNF). According to the shape of the input acceptance condition, we choose the suitable (shorter) normal form and perform the merging algorithm on it.

By choosing the CNF over DNF in the cases where CNF is more natural (shorter), we prevent the formula to contain some acceptance marks more than once (at least, we do not create them by conversion of ψ to DNF). This way, we prevent adding new acceptance marks to the formula when resolving conflicts of acceptance marks which leads to better results. If the original formula contains an acceptance mark that is present in more than one disjunct, then by ordering the disjuncts in every $\overline{\psi_i}$, the *linear sum assignment* returns more convenient pairing. Meaning that if *linear sum assignment* finds two equal assignments, it respects the order and the disjuncts with the acceptance mark that occurs more than once are paired with disjuncts that also contain acceptance mark that occurs more than once.

Example 2. Consider an automaton $\mathcal{A}$ in Figure 1. In this example, we demonstrate the simplification procedure described in Section 3.1 and obtain simplified acceptance conditions ψ_i for each SCC S_i. The simplified acceptance conditions $\overline{\psi_i}$ are displayed in the set-format we introduced earlier because we make use of it afterward during the merge. Then we merge these formulae into the acceptance condition of the simplified automaton $\mathcal{A}'$. We enumerate the SCCs S_1, S_2, S_3. (The indices correspond with the numbers inside the states in Figure 1.) And we assign an accepting condition ψ_1 to S_1, ψ_2 to S_2 and ψ_3 to S_3.

1. We simplify ψ_1 according to the placement of acceptance marks on the edges in S_1. At first we notice, that the formula $\overline{\psi_1}$ contains acceptance marks that are not present on the edges of S_1. Therefore modify $\overline{\psi_1}$ as described in Subsection 3.1. We substitute *true* for Fin❸ and *false* for Inf❶ and Inf⓿. (This substitution deletes the whole disjunct D_1.) Then we notice that the acceptance marks are in a position which allows us to perform the simplification described in first part of subsection 3.1.4. We substitute *true* for Inf❹ and obtain the final form of $\overline{\psi_1} = \{\{\mathsf{Fin}❷\}\}$.
2. Similarly as in previous case, we notice that acceptance marks ❷, ❸ and ❹ are not present on the edges of S_2. Therefore we substitute *true* for both Fin❷ and Fin❸ and *false* for Inf❹ (and thus remove the whole disjunct D_2). Then we use the procedure described in Subsection 3.1 and remove Inf❶ from the disjunct D_1 and obtain $\overline{\psi_2} = \{\{\mathsf{Inf}⓿\}\}$.
3. Finally, we yet again remove marks that are not present in S_3 and perform the simplification described in subsection 3.1.3. We substitute *true* for Fin❷ and Fin❸ and add new term Fin❺ into D_2 and place ❺ on the edges that are labeled with ❷ or ❸. The simplified acceptance condition is $\overline{\psi_3} = \{\{\mathsf{Fin}❺, \mathsf{Inf}❹\}\}$.

Now we proceed to the merge of the formulae of the SCCs ψ_i. As a base of the new acceptance condition ψ' we choose the formula ψ_3 because it has the highest number of disjuncts compared to the other formulae ψ_i, so $\overline{\psi'} = \{\{\mathsf{Fin}❺, \mathsf{Inf}❹\}\}$.

Now we merge ψ_1 and ψ_2 with ψ'. The order is in this particular case irrelevant because $\overline{\psi_1}$ has the same number of disjuncts as $\overline{\psi_2}$ (and both disjuncts have the same number of conjuncts). So we merge $\overline{\psi_1}$ at first because of its lower index. Since both $\overline{\psi'}$ and $\overline{\psi_1}$ have only one disjunct, the only possible pairing that *linear sum assignment* can give us is $(0, 0)$, meaning that {Fin②} is merged with {Fin⑤, Inf④}. Then we need to map an unused conjunct Fin⑤ to a conjunct of the same type which is Fin②. Similarly in case of $\overline{\psi_2}$ we map Inf④ on Inf⓪. This way we produce $\mathcal{A}'$ displayed in Figure 2 with the acceptance condition Fin⑤ ∧ Inf④. Observe that the automaton $\mathcal{A}'$ is not equivalent to the original $\mathcal{A}$. For example in Figure 2 is the highlighted loop not accepting but in the original automaton in Figure 1 the loop accepting is. To reach the equivalence between the two automata, we need to *restore equivalence* which is described in the next section.

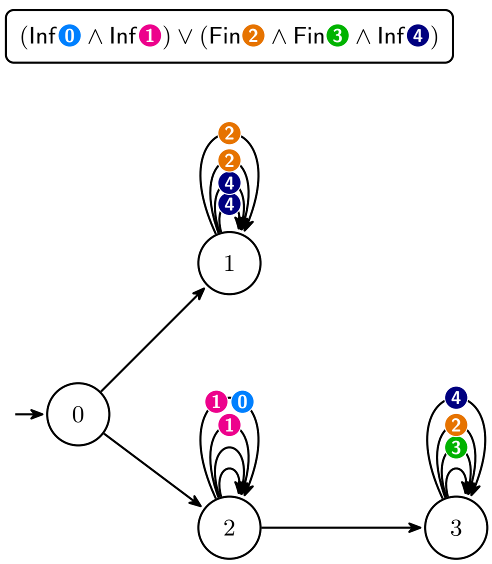

Fig. 1. TELA before simplification Level 1

3.3 Restoring equivalence

Replacing the input acceptance condition with ψ' in $\mathcal{A}$ might change the language, so it is no longer equivalent to the language recognized by the original automaton $\mathcal{A}$. By now, each SCC S_i is adjusted to the acceptance condition ψ_i which might be different than ψ'. Therefore, for each SCC we need to restore equivalence with the new acceptance condition ψ'. The idea is that the only terms of ψ' that can affect accepting runs on SCC S_i are the ones that have

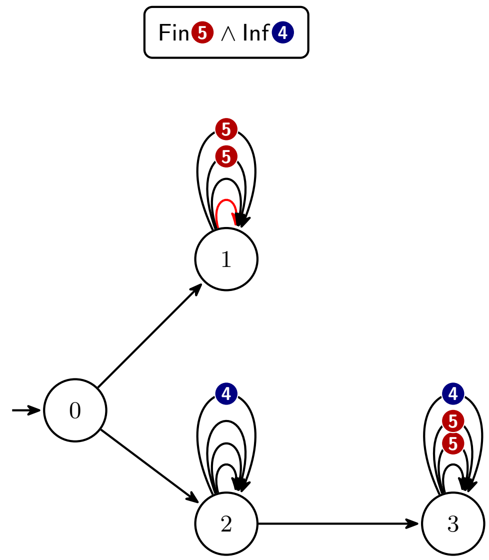

Fig. 2. TELA after simplification and merging in Level 1

been mapped on some terms in ψ_i. From now on, we refer to them as *used terms*. We need to make sure that the other terms (the ones that have not been mapped on any term of ψ_i) can't affect any accepting run of $\mathcal{A}'$. From now on, we call these terms *unused terms*. We influence the evaluation of the *unused terms* in the following way:

- If we want to make term in the Fin form always *false*, we place it's acceptance mark on every edge of the SCC.
- If we want to make term in the Inf form always *true*, we place it's acceptance mark on every edge of the SCC.

Keep in mind that by now, any acceptance mark that is not in ψ' is not present on any edge of $\mathcal{A}'$. Therefore any Fin term is already always *true* if its acceptance mark is not present on any edge of the SCC. Dually, Inf term is already always *false* if it is not present on any edge of the SCC. Now we distinguish the cases when ψ' is in DNF or ψ' is in CNF.

- If ψ' is in DNF, we need to make *false* every disjunct that does not contain any *used term*. Further, every *unused term* that occurs in the same disjunct as any *used term* needs to be made *true*.
- Dually, if ψ' is in CNF, every conjunct that does not contain any *used term* needs to be made *true* and every *unused term* that occurs in a conjunct with any *used term* needs to be made *false*.

Example 3. Consider the example from the Figure 2, where the DNF acceptance condition $\psi' = (\mathsf{Fin}\,5 \vee \mathsf{Inf}\,4)$ and the acceptance condition of SCC S_1 $\psi_1 =$

(Fin❷). The term Fin❺ is mapped to Fin❷. Then we need to enforce that the *unused term* Inf❹ has no effect on the evaluation of ψ' by the run with the infinite suffix contained within the SCC S_1. Otherwise the result will not be equivalent (see the highlited loop in Figure 2). Therefore we make term Inf❹ *true* by adding ❹ on every edge of the SCC S_i. The result can be seen in Figure 3.

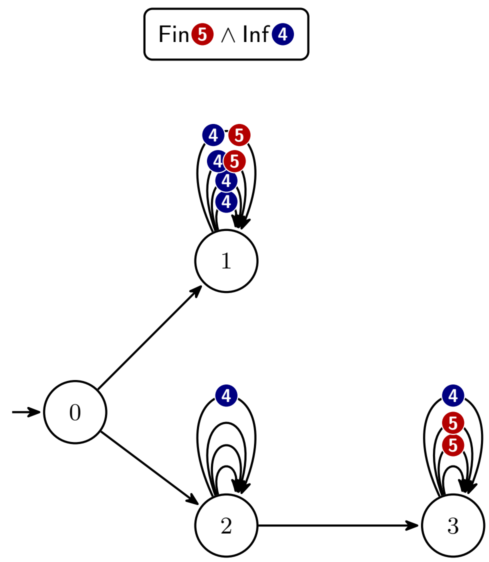

Fig. 3. Equivalent TELA after simplification Level 1

4 QBF-based reduction

– obecny popis konstrukce formule
– (level 3)
– level 4

5 Implementation and experimental evaluation

6 Conclusions