A Bounded Model Checker for Three-Valued Abstractions of Software Systems – Proofs

Nils Timm, Stefan Gruner, and Matthias Harvey

Department of Computer Science, University of Pretoria, South Africa
{ntimm,sgruner}@cs.up.ac.za

In the following we present the proof of Theorem 1 from the paper "A Bounded Model Checker for Three-Valued Abstractions of Software Systems", submitted to the 19th Brazilian Symposium on Formal Methods. The theorem that we prove is the following:

Theorem 1. *Let M be a three-valued Kripke structure representing the state space of an abstracted concurrent system Sys, let ψ be an LTL formula and $b \in \mathbb{N}$ Then:*

$$[M \models_{E,b} \psi] \;\equiv\; \begin{cases} \mathit{true} & \mathit{iff} \;\; \mathrm{SAT}([\![Sys,\psi]\!]_b[\bot/\mathit{false}]) = \mathit{true} \\ \mathit{false} & \mathit{iff} \;\; \mathrm{SAT}([\![Sys,\psi]\!]_b[\bot/\mathit{true}]) = \mathit{false} \\ \bot & \mathit{else} \end{cases}$$

Proof of Theorem 1.
We prove Theorem 1 by showing that for each b-bounded path π in M there exits an assignment $\alpha_\pi : \mathit{Atoms}_{[0,b]} \to \{\mathit{true}, \mathit{false}\}$ that exactly characterises π in $[\![Sys]\!]_b$, i.e. the transition values along π and $\alpha_\pi([\![Sys]\!]_b)$ are identical and the labellings along π and $\alpha_\pi([\![Sys]\!]_b)$ are identical as well (Lemma 1). Moreover, we show that the evaluation of an LTL property ψ on ψ yields the same result as $\alpha_\pi([\![\psi]\!]_b)$ (Lemma 2). Note that the encoding of states (Definitions 7 and 8) always yields a conjunction of literals. Hence, for each state encoding $\mathit{enc}(\langle l,s\rangle)_k$ (resp. $\mathit{enc}(l)_k$ and $\mathit{enc}(s)_k$) there exists exactly one assignment to its atoms that makes the encoding *true*. We denote such an assignment characterising a state $\langle l,s\rangle$ by $\alpha_{\langle l,s\rangle}$ (resp. α_l and α_s):

Definition 13. (Assignments Characterising States) *Let $\langle l,s\rangle$ be a state of a Kripke structure M and let $k \in \mathbb{N}$ arbitrary but fixed. The encodings $\mathit{enc}(l)_k$ and $\mathit{enc}(s)_k$ are conjunctions of literals. Hence, there exists exactly one satisfying assignment to its atoms. We denote these satisfying assignment as α_l resp. α_s and we have by definition that the following holds:*

$$\alpha_l(\mathit{enc}(l)_k) = \mathit{true} \text{ and } \alpha_s(\mathit{enc}(s)_k) = \mathit{true}.$$

Such assignments can be generalised to pairs of states, i.e.

$$\begin{aligned} \alpha_{l,l'}(\mathit{enc}(l)_k \wedge \mathit{enc}(l')_{k+1}) &= \alpha_l(\mathit{enc}(s)_k) \wedge \alpha_{l'}(\mathit{enc}(l')_{k+1}) = \mathit{true}, \\ \alpha_{s,s'}(\mathit{enc}(s)_k \wedge \mathit{enc}(s')_{k+1}) &= \alpha_s(\mathit{enc}(s)_k) \wedge \alpha_{s'}(\mathit{enc}(s')_{k+1}) = \mathit{true} \end{aligned}$$

where we assume that such a pair is encoded with consecutive position values k and $k+1$.

This notion of assignments characterising states can be straightforwardly transferred to sub-states (e.g. $\alpha_l(\mathit{enc}(l_i)_k) = \mathit{true}$ with $l = (l_1, \ldots, l_i, \ldots, l_n)$) and to compound states (e.g $\alpha_{\langle l,s\rangle}(\mathit{enc}(l)_k) = \mathit{true}$ and $\alpha_{\langle l,s\rangle}(\mathit{enc}(s)_k) = \mathit{true}$). Our fist step towards proving Theorem 1 is to establish a relation between states and transitions of an explicit Kripke structure, and assignments characterising states and transitions in a corresponding encoding. For this we prove Lemma 1:

Lemma 1. *Let $M = (S, \langle l_0, s_0\rangle, R, L)$ over AP be a three-valued Kripke structure representing the state space of an abstracted concurrent system Sys and let $k \in \mathbb{N}$. Moreover, let $\langle l,s\rangle$ and $\langle l',s'\rangle$ be a pair of arbitrary states of M and let p_j and $(\mathit{loc}_i = \hat{l}_i)$ be atomic predicates in AP. Then the following equivalences hold:*

(A) $\alpha_{\langle l_0, s_0\rangle}(Init_0) \;\equiv\; true$

(B) $R(\langle l, s\rangle, \langle l', s'\rangle) \;\equiv\; \alpha_{\langle l,s\rangle,\langle l',s'\rangle}(Trans_{k,k+1})$

(C) $L(\langle l, s\rangle, p_j) \;\equiv\; \alpha_{\langle l,s\rangle}(enc(p_j)_k)$

(D) $L(\langle l, s\rangle, (loc_i = \hat{l}_i)) \;\equiv\; \alpha_{\langle l,s\rangle}(enc(\hat{l}_i)_k)$

Proof of Lemma 1.
(A) We have that $Init_0 = enc(\langle l_0, s_0\rangle)_0$. Moreover, $\alpha_{\langle l_0,s_0\rangle}(enc(\langle l_0, s_0\rangle)_0) \equiv true$ holds by definition of $\alpha_{\langle l_0,s_0\rangle}$ (Definition 13), which completes this part of the proof.

(B) According to Definition 7 in [1], $R(\langle l, s\rangle, \langle l', s'\rangle)$ can be rewritten as

$$\bigvee_{i=1}^{n}(\delta_i(l_i, l'_i) \wedge \bigwedge_{i'\neq i} l_{i'} = l'_{i'} \wedge s(choice(a, b)) \wedge \bigwedge_{j=1}^{m} s'(p_j) = s(choice(a_j, b_j)))$$

assuming that $l = (l_1, \dots, l_n)$, $l' = (l'_1, \dots, l'_n)$ and $\tau_i(l_i, l'_i) \;=\; assume(choice(a, b)) : p_1 := choice(a_1, b_1), \dots, p_m := choice(a_m, b_m)$. In order to prove Part (B) of the Lemma we show that the following equivalences hold:

1. $\delta_i(l_i, l'_i) \wedge \bigwedge_{i'\neq i} l_{i'} = l'_{i'} \;\equiv\; \bigvee_{(\hat{l}_i, \hat{l}'_i)\in\delta_i}(\alpha_l(enc(\hat{l}_i)_k) \wedge \alpha_{l'}(enc(\hat{l}'_i)_{k+1}) \wedge \bigwedge_{i'\neq i}\bigwedge_{j=1}^{d_{i'}} \alpha_l(\hat{l}_{i'}[j]_k) \leftrightarrow \alpha_{l'}(\hat{l}'_{i'}[j]_{k+1}))$

2. $s(choice(a, b)) \;\equiv\; \alpha_s(enc(choice(a, b))_k)$

3. $s'(p_j) = s(choice(a_j, b_j)) \;\equiv\; \alpha_{s,s'}((enc(a_j)_k \wedge enc(p_j = true)_k + 1) \vee (enc(b_j)_k \wedge enc(p_j = false)_k + 1) \vee (enc(\neg a_j \wedge \neg b_j)_k[\bot/true] \wedge enc(p_j = \bot)_k + 1))$

We start with proving the '⇒'-direction of Equivalence 1. From the first part of the premise we can derive the following:

$\delta_i(l_i, l'_i) = true$,
(Premise)
$\alpha_l(enc(l_i)_k) = true$,
$\alpha_{l'}(enc(l'_i)_{k+1}) = true$
(Definition 13)
$\Rightarrow (\bigvee_{(\hat{l}_i, \hat{l}'_i)\in\delta_i} \alpha_l(enc(\hat{l}_i)_k) \wedge \alpha_{l'}(enc(\hat{l'}_i)_{k+1})) = true$
(Deduction)

Moreover, from the second part of the premise we can derive the following:

$\bigwedge_{i' \neq i}(l_{i'} = l'_{i'})$
(Premise)

$\Rightarrow \bigwedge_{i' \neq i}(enc(l_{i'})_k[k/k+1] = enc(l'_{i'})_{k+1})$
with $enc(l_{i'})_k = \bigwedge_{j=1}^{d_{i'}}((l_{i'}[j]_k \wedge b_{l_{i'}}(j)) \vee (\neg l_{i'}[j]_k \wedge \neg b_{l_{i'}}(j)))$
and $enc(l'_{i'})_{k+1} = \bigwedge_{j=1}^{d_{i'}}((l'_{i'}[j]_{k+1} \wedge b_{l'_{i'}}(j)) \vee (\neg l'_{i'}[j]_{k+1} \wedge \neg b_{l'_{i'}}(j)))$
(Definition 7)

$\Rightarrow \bigwedge_{i' \neq i}(\bigwedge_{j=1}^{d_{i'}}(\alpha_l(l_{i'}[j]_k) = \alpha_{l'}(l'_{i'}[j]_{k+1}))) = true$
(Definition 13)

$\Rightarrow \bigwedge_{i' \neq i}(\bigwedge_{j=1}^{d_{i'}}(\alpha_l(l_{i'}[j]_k) \leftrightarrow \alpha_{l'}(l'_{i'}[j]_{k+1}))) = true$
(Equivalent transformation)

Together we get $\delta_i(l_i, l'_i) \wedge \bigwedge_{i' \neq i} l_{i'} = l'_{i'} \;\Rightarrow\; \bigvee_{(\hat{l}_i, \hat{l}'_i) \in \delta_i}(\alpha_l(enc(\hat{l}_i)_k) \wedge \alpha_{l'}(enc(\hat{l}'_i)_{k+1}) \wedge \bigwedge_{i' \neq i} \bigwedge_{j=1}^{d_{i'}} \alpha_l(\hat{l}_{i'}[j]_k) \leftrightarrow \alpha_{l'}(\hat{l}'_{i'}[j]_{k+1}))$. Next, we prove the '$\Leftarrow$'-direction by showing that if the left side of the equivalence evaluates to *false* then also the right side evaluates to *false*. A *false* on the left side means $\delta_i(l_i, l'_i) = false$ or $\bigvee_{i' \neq i}(l_{i'} \neq l'_{i'})$. We show that in both cases the right side will evaluate to *false* as well. We start with the first case:

$\delta_i(l_i, l'_i) = false$ (which is equivalent to $(l_i, l'_i) \notin \delta_i$),
(Premise)
$\alpha_l(enc(l_i)_k) = true \wedge \bigwedge_{\hat{l}_i \neq l_i} \alpha_l(enc(\hat{l}_i)_k) = false$,
$\alpha_{l'}(enc(l'_i)_{k+1}) = true \wedge \bigwedge_{\hat{l'}_i \neq l'_i} \alpha'_l(enc(\hat{l'}_i)_{k+1}) = false$,
(Definition 13)

$\Rightarrow (\bigvee_{(\hat{l}_i, \hat{l}'_i) \in \delta_i} \alpha_l(enc(\hat{l}_i)_k) \wedge \alpha_{l'}(enc(\hat{l'}_i)_{k+1})) = false$
(Deduction)

Next, we consider the second case:

$\bigvee_{i' \neq i}(l_{i'} \neq l'_{i'})$
(Premise)

$\Rightarrow \bigvee_{i' \neq i}(enc(l_{i'})_k[k/k+1] \neq enc(l'_{i'})_{k+1})$
with $enc(l_{i'})_k = \bigwedge_{j=1}^{d_{i'}}((l_{i'}[j]_k \wedge b_{l_{i'}}(j)) \vee (\neg l_{i'}[j]_k \wedge \neg b_{l_{i'}}(j)))$
and $enc(l'_{i'})_{k+1} = \bigwedge_{j=1}^{d_{i'}}((l'_{i'}[j]_{k+1} \wedge b_{l'_{i'}}(j)) \vee (\neg l'_{i'}[j]_{k+1} \wedge \neg b_{l'_{i'}}(j)))$
(Definition 7)

$\Rightarrow \bigvee_{i' \neq i}(\bigvee_{j=1}^{d_{i'}}(\alpha_l(l_{i'}[j]_k) \neq \alpha_{l'}(l'_{i'}[j]_{k+1}))) = true$
(Definition 13)

$\Rightarrow \bigwedge_{i' \neq i}(\bigwedge_{j=1}^{d_{i'}}(\alpha_l(l_{i'}[j]_k) \leftrightarrow \alpha_{l'}(l'_{i'}[j]_{k+1}))) = false$
(Equivalent transformation)

Hence, $\delta_i(l_i, l'_i) \wedge \bigwedge_{i' \neq i} l_{i'} = l'_{i'} \;\Leftarrow\; \bigvee_{(\hat{l}_i, \hat{l}'_i) \in \delta_i}(\alpha_l(enc(\hat{l}_i)_k) \wedge \alpha_{l'}(enc(\hat{l}'_i)_{k+1}) \wedge \bigwedge_{i' \neq i} \bigwedge_{j=1}^{d_{i'}} \alpha_l(\hat{l}_{i'}[j]_k) \leftrightarrow \alpha_{l'}(\hat{l}'_{i'}[j]_{k+1}))$ holds as well, which completes the proof of Equivalence 1.

Next we prove Equivalence 2. We show that $s(choice(a, b)) \equiv \alpha_s(enc(choice(a, b))_k)$ holds. We distinguish the following cases:

2.1. $s(choice(a, b)) = true \equiv \alpha_s(enc(choice(a, b))_k) = true$
2.2. $s(choice(a, b)) = false \equiv \alpha_s(enc(choice(a, b))_k) = false$

2.3. $s(choice(a, b)) = \perp \;\equiv\; \alpha_s(enc(choice(a, b))_k) = \perp$

In all three cases we start with the transformation of $enc(choice(a, b))_k$. The following equivalence holds: $enc(choice(a, b))_k \equiv enc((a \vee \neg b) \wedge (a \vee b \vee \perp))_k \equiv (enc(a)_k \vee enc(\neg b)_k) \wedge (enc(a)_k \vee enc(b)_k \vee \perp)$ (Definition 9). Case 2.1: We show $s(choice(a, b)) = true \;\equiv\; \alpha_s((enc(a)_k \vee enc(\neg b)_k) \wedge (enc(a)_k \vee enc(b)_k \vee \perp)) = true$. From the semantics of the *choice* expression we get $s(choice(a, b)) = true \;\equiv\; s(a) = true$. Hence, it is sufficient to show that $s(a) = true \;\equiv\; \alpha_s((enc(a)_k \vee enc(\neg b)_k) \wedge (enc(a)_k \vee enc(b)_k \vee \perp)) = true$ holds. The logical expression a is defined over *Pred* and we can assume that a has been transferred into negation normal form. Now Case 2.1 can be proven by induction over the structure of a. We distinguish the following cases:

2.1.1. $a = p_i$ with $p_i \in Pred$,

2.1.2. $a = \neg p_i$ with $p_i \in Pred$,

2.1.3. $a = e \vee e'$ with e, e' logical expressions in NNF over *Pred*,

2.1.4. $a = e \wedge e'$ with e, e' logical expressions in NNF over *Pred*.

Case 2.1.1:

$$s(p_j) = true \;\equiv\; \alpha_s(enc(choice(p_j, b))_k) = true$$

We start with the '⇒'-direction. Hence, we have to show that $s(p_j) = true$ implies $\alpha_s(enc(choice(p_j, b))_k) = true$. For this, we firstly derive a fact from the premise $s(p_j) = true$, which we can then use for proving that $\alpha_s(enc(choice(p_j, b))_k) = true$ holds.

$$\begin{array}{lll}
 & s(p_j) = true, & \text{(Premise)} \\
 & \alpha_s(enc(s)_k) = true & \text{(Definition 13)} \\
\Rightarrow & s(p_j) = true, & \\
 & \alpha_s(\bigwedge_{p \in Pred} enc(p = s(p))_k) = true & \text{(Definition 8)} \\
\Rightarrow & \alpha_s(enc(p_j = true)_k) = true & \text{(Deduction)} \\
\Rightarrow & \alpha_s(\neg p_j[u]_k \wedge p_j[t]_k) = true & \text{(Definition 8)} \\
\Rightarrow & \alpha_s(p_j[u]_k) = false, & \text{(Deduction)} \\
 & \alpha_s(p_j[t]_k) = true &
\end{array}$$

Hence, we have proven that $s(p_j) = true$ implies $\alpha_s(p_j[u]_k) = false$ and $\alpha_s(p_j[t]_k) = true$, which we denote as *Fact 1*. Now we can prove that $s(p_j) = true$ also implies $\alpha_s(enc(choice(p_j, b))_k) = true$. For this, we transform $\alpha_s(enc(choice(p_j, b))_k)$ and make use of Fact 1:

$$\begin{aligned}
& \alpha_s(enc(choice(p_j, b))_k) \\
\equiv\ & \alpha_s(enc((p_j \vee \neg b) \wedge (p_j \vee b \vee \bot))_k) \\
& \text{(Definition 9)} \\
\equiv\ & (\alpha_s(enc(p_j)_k) \vee \alpha_s(enc(\neg b)_k)) \wedge (\alpha_s(enc(p_j)_k) \vee \alpha_s(enc(b)_k) \vee \bot) \\
& \text{(Definition 9)} \\
\equiv\ & (\alpha_s((p_j[u]_k \wedge \bot) \vee (\neg p_j[u]_k \wedge p_j[t]_k)) \vee \alpha_s(enc(\neg b)_k)) \\
\wedge\ & (\alpha_s((p_j[u]_k \wedge \bot) \vee (\neg p_j[u]_k \wedge p_j[t]_k)) \vee \alpha_s(enc(b)_k) \vee \bot) \\
& \text{(Definition 9)} \\
\equiv\ & ((\alpha_s(p_j[u]_k) \wedge \bot) \vee (\neg\alpha_s(p_j[u]_k) \wedge \alpha_s(p_j[t]_k)) \vee \alpha_s(enc(\neg b)_k)) \\
\wedge\ & ((\alpha_s(p_j[u]_k) \wedge \bot) \vee (\neg\alpha_s(p_j[u]_k) \wedge \alpha_s(p_j[t]_k)) \vee \alpha_s(enc(b)_k) \vee \bot) \\
& \text{(Equivalent transformation)} \\
\equiv\ & ((\mathit{false} \wedge \bot) \vee (\mathit{true} \wedge \mathit{true}) \vee \alpha_s(enc(\neg b)_k)) \\
\wedge\ & ((\mathit{false} \wedge \bot) \vee (\mathit{true} \wedge \mathit{true}) \vee \alpha_s(enc(b)_k) \vee \bot) \\
& \text{(Fact 1)} \\
\equiv\ & (\mathit{true} \vee \alpha_s(enc(\neg b)_k)) \wedge (\mathit{true} \vee \alpha_s(enc(b)_k) \vee \bot) \\
& \text{(Equivalent transformation)} \\
\equiv\ & \mathit{true} \\
& \text{(Equivalent transformation)}
\end{aligned}$$

Consequently, $s(p_j) = \mathit{true} \;\Rightarrow\; \alpha_s(enc(choice(p_j, b))_k) = \mathit{true}$ holds. Next we prove the '$\Leftarrow$'-direction. Hence, we have to show that $\alpha_s(enc(choice(p_j, b))_k) = \mathit{true}$ implies $s(p_j) = \mathit{true}$. For this, we firstly derive a fact from the premise $\alpha_s(enc(choice(p_j, b))_k) = \mathit{true}$, which we can then use for proving that $s(p_j) = \mathit{true}$ holds.

$$\begin{aligned}
& \alpha_s(enc(choice(p_j, b))_k) = \mathit{true} \\
\equiv\ & (\alpha_s((p_j[u]_k \wedge \bot) \vee (\neg p_j[u]_k \wedge p_j[t]_k)) \vee \alpha_s(enc(\neg b)_k)) \\
\wedge\ & (\alpha_s((p_j[u]_k \wedge \bot) \vee (\neg p_j[u]_k \wedge p_j[t]_k)) \vee \alpha_s(enc(b)_k) \vee \bot) = \mathit{true} \\
& \text{(Compare transformations for '}\Rightarrow\text{'-direction)} \\
\Rightarrow\ & \alpha_s((p_j[u]_k \wedge \bot) \vee (\neg p_j[u]_k \wedge p_j[t]_k)) = \mathit{true} \\
& \text{(Fact that } \alpha_s(enc(b)_k) \text{ and } \alpha_s(enc(\neg b)_k) \text{ are complementary)} \\
\Rightarrow\ & \alpha_s(p_j[u]_k) = \mathit{false}, \\
& \alpha_s(p_j[t]_k) = \mathit{true} \\
& \text{(Deduction)}
\end{aligned}$$

Hence, we have proven that $\alpha_s(enc(choice(p_j, b))_k) = \mathit{true}$ implies $\alpha_s(p_j[u]_k) = \mathit{false}$ and $\alpha_s(p_j[t]_k) = \mathit{true}$, which we denote as *Fact 2*. We now prove that this also implies $s(p_j) = \mathit{true}$. For this, we firstly show that $\alpha_s(enc(p_j = s(p_j))_k) = \mathit{true}$ holds:

$$\begin{aligned}
& \alpha_s(enc(s)_k) = \mathit{true} \\
& \text{(Definition 13)} \\
\equiv\ & \alpha_s(\textstyle\bigwedge_{p \in Pred} enc(p = s(p))_k) = \mathit{true} \\
& \text{(Definition 8)} \\
\Rightarrow\ & \alpha_s(enc(p_j = s(p_j))_k) = \mathit{true} \\
& \text{(Deduction)}
\end{aligned}$$

Hence, we have shown that $\alpha_s(enc(p_j = s(p_j))_k) = true$, which we denote as *Fact 3*. Now we can prove that from Fact 2 and Fact 3 we can deduce that $s(p_j) = true$ holds. We have that $s(p_j) \in \{true, \perp, false\}$. We now show that only $s(p_j) = true$ is conform with Fact 3: Let $s(p_j) = true$. Then $enc(p_j = s(p_j))_k = \neg p_i[u]_k \wedge p_i[t]_k$ (Definition 8). Combining this with Fact 2 gives us $\alpha_s(enc(p_j = s(p_j))_k) = true$, which is conform with Fact 3. Let $s(p_j) = false$. Then $enc(p_j = s(p_j))_k = \neg p_i[u]_k \wedge \neg p_i[t]_k$ (Definition 8). Combining this with Fact 2 gives us $\alpha_s(enc(p_j = s(p_j))_k) = false$, which is a contradiction to Fact 3. Let $s(p_j) = \perp$. Then $enc(p_j = s(p_j))_k = p_i[u]_k$ (Definition 8). Combining this with Fact 2 gives us $\alpha_s(enc(p_j = s(p_j))_k) = false$, which is a contradiction to Fact 3. Consequently, $s(p_j) = true$ follows from Fact 2 and Fact 3. Hence, we have proven that also the '$\Leftarrow$'-direction $\alpha_s(enc(choice(p_j, b))_k) = true \;\Rightarrow\; s(p_j) = true$ holds. Altogether we get

$$s(p_j) = true \;\equiv\; \alpha_s(enc(choice(p_j, b))_k) = true$$

which completes the proof of Case 2.1.1.

The proof of Case 2.1.2 is analogous to the proof of Case 2.1.1. Thus, next we consider Case 2.1.3:

$$s(e \vee e') = true \;\equiv\; \alpha_s(enc(choice(e \vee e', b))_k) = true$$

The following equivalences hold:

$$\begin{array}{ll}
 & s(e \vee e') = true \\
 & \text{(Premise)} \\
\equiv & s(e) = true \\
 & \vee\, s(e') = true \\
 & \text{(Equivalent transformation)} \\
\equiv & \alpha_s(enc(choice(e, b))_k) = true \\
 & \vee\, \alpha_s(enc(choice(e', b))_k) = true \\
 & \text{(Induction)} \\
\equiv & \alpha_s(enc((e \vee \neg b) \wedge (e \vee b \vee \perp))_k) = true \\
 & \vee\, \alpha_s(enc((e' \vee \neg b) \wedge (e' \vee b \vee \perp))_k) = true \\
 & \text{(Definition 9)} \\
\equiv & (\alpha_s(enc(e)_k) \vee \alpha_s(enc(\neg b)_k)) \wedge (\alpha_s(enc(e)_k) \vee \alpha_s(enc(b)_k) \vee \perp) = true \\
 & \vee\, (\alpha_s(enc(e')_k) \vee \alpha_s(enc(\neg b)_k)) \wedge (\alpha_s(enc(e')_k) \vee \alpha_s(enc(b)_k) \vee \perp) = true \\
 & \text{(Definition 9)} \\
\equiv & \alpha_s(enc(e)_k) = true \\
 & \vee\, \alpha_s(enc(e')_k) = true \\
 & \text{(Fact that } \alpha_s(enc(b)_k) \text{ and } \alpha_s(enc(\neg b)_k) \text{ are complementary)} \\
\equiv & \alpha_s(enc(e \vee e')_k) = true \\
 & \text{(Definition 9, Equivalent transformation)} \\
\equiv & (\alpha_s(enc(e \vee e')_k) \vee \alpha_s(enc(\neg b)_k)) \wedge (\alpha_s(enc(e \vee e')_k) \vee \alpha_s(enc(b)_k) \vee \perp) = true \\
 & \text{(Equivalent transformation, Fact that } \alpha_s(enc(b)_k) \text{ and } \alpha_s(enc(\neg b)_k) \text{ are complementary)} \\
\equiv & \alpha_s(enc((e \vee e' \vee \neg b) \wedge (e \vee e' \vee b \vee \perp))_k) = true \\
 & \text{(Definition 9, Equivalent transformation)} \\
\equiv & \alpha_s(enc(choice(e \vee e', b))_k) = true \\
 & \text{(Definition 9)}
\end{array}$$

Hence,

$$s(e \vee e') = true \;\equiv\; \alpha_s(enc(choice(e \vee e', b))_k) = true$$

which completes the proof of Case 2.1.3. The proof of Case 2.1.4 is analogous to the proof of Case 2.1.3. Hence, we have completed the proof of Case 2.1. The proofs of Case 2.2 and Case 2.3 are again analogous to the proof of Case 2.1. We only have to start with different premises: $(s(a) = \mathit{false} \vee s(a) = \bot) \wedge (s(b) = \mathit{true})$ (Case 2.2) resp. $(s(a) = \mathit{false} \vee s(a) = \bot) \wedge (s(b) = \mathit{false} \vee s(b) = \bot)$ (Case 2.3) and show that $\alpha_s(enc(choice(a, b))_k)$ is equivalent to *false* (Case 2.2) resp. $\bot$ (Case 2.3).

For the proof of Case 3 we have to show that the following equivalence holds:

$$\begin{aligned}
&\textstyle\bigwedge_{j=1}^{m} \; (\; s'(p_j) = s(choice(a_j, b_j))) \\
\equiv &\textstyle\bigwedge_{j=1}^{m} \; (\quad (\alpha_s(enc(a_j)_k) \wedge \alpha_{s'}(enc(p_j = \mathit{true})_{k+1})) \\
&\qquad \vee (\alpha_s(enc(b_j)_k) \wedge \alpha_{s'}(enc(p_j = \mathit{false})_{k+1})) \\
&\qquad \vee (\alpha_s(enc(\neg a_j \wedge \neg b_j)_k[\bot/\mathit{true}]) \wedge \alpha_{s'}(enc(p_j = \bot)_{k+1})))
\end{aligned}$$

For this it is sufficient to show that

$$\begin{aligned}
&s'(p_j) = s(choice(a_j, b_j)) \\
\equiv \quad & \quad (\alpha_s(enc(a_j)_k) \wedge \alpha_{s'}(enc(p_j = \mathit{true})_{k+1})) \\
& \vee (\alpha_s(enc(b_j)_k) \wedge \alpha_{s'}(enc(p_j = \mathit{false})_{k+1})) \\
& \vee (\alpha_s(enc(\neg a_j \wedge \neg b_j)_k[\bot/\mathit{true}]) \wedge \alpha_{s'}(enc(p_j = \bot)_{k+1}))
\end{aligned}$$

holds for an arbitrary but fixed $j \in \{1, \ldots, m\}$. The following table lists the cases that we have to consider (Columns 1 and 2). Moreover, is shows the result of the corresponding equation (Column 3).

$s'(p_j)$	$s(choice(a_j, b_j))$	$s'(p_j) = s(choice(a_j, b_j))$
true	*true*	*true*
false	*false*	*true*
$\bot$	$\bot$	*true*
true	$\bot$	$\bot$
$\bot$	*true*	$\bot$
false	$\bot$	$\bot$
$\bot$	*false*	$\bot$
true	*false*	*false*
false	*true*	*false*

Fig. 1. Proof Cases.

Hence, we have to show that in all cases the result of the equation is equivalent to the result of the encoding under the assignments α_s and $\alpha_{s'}$. For each case the proof is similar. Here we show the proof of the most complex case

$$s'(p_j) = \bot, \; s(choice(a_j, b_j)) = \bot, \; (s'(p_j) = s(choice(a_j, b_j))) \equiv \mathit{true}$$

i.e. we show that under this premise

$$\begin{aligned}
& \quad (\alpha_s(enc(a_j)_k) \wedge \alpha_{s'}(enc(p_j = \mathit{true})_{k+1})) \\
& \vee (\alpha_s(enc(b_j)_k) \wedge \alpha_{s'}(enc(p_j = \mathit{false})_{k+1})) \\
& \vee (\alpha_s(enc(\neg a_j \wedge \neg b_j)_k[\bot/\mathit{true}]) \wedge \alpha_{s'}(enc(p_j = \bot)_{k+1}))
\end{aligned}$$

is equivalent to *true*. The proof goes by induction over the structure of a_j and b_j. We show the case $a_j = q$ and $b_j = r$ with $q, r \in \mathit{Pred}$. The proof of the other cases via induction is based on the same argumentation as the proof of Case 2 (e.g. compare proof of Case 2.1.3). We start with the '$\Rightarrow$'-direction. For this, we firstly derive a fact with regard to α_s and $\alpha_{s'}$ from the premise.

$$
\begin{array}{lll}
 & s(\mathit{choice}(q,r)) = \bot, & \text{(Premise)} \\
 & s'(p_j) = \bot, & \text{(Premise)} \\
 & \alpha_s(\mathit{enc}(s)_k) = \mathit{true}, & \text{(Definition 13)} \\
 & \alpha_{s'}(\mathit{enc}(s)_{k+1}) = \mathit{true} & \text{(Definition 13)} \\
\Rightarrow & s(q) = \mathit{false} \;\vee\; s(q) = \bot, & \text{(Definition 9)} \\
 & s(r) = \mathit{false} \;\vee\; s(r) = \bot, & \text{(Definition 9)} \\
 & s'(p_j) = \bot, & \\
 & \alpha_s(\bigwedge_{p \in \mathit{Pred}} \mathit{enc}(p = s(p))_k) = \mathit{true}, & \text{(Definition 8)} \\
 & \alpha_{s'}(\bigwedge_{p \in \mathit{Pred}} \mathit{enc}(p = s'(p))_{k+1}) = \mathit{true} & \text{(Definition 8)} \\
\Rightarrow & \alpha_s(\mathit{enc}(q = \mathit{false})_k) = \mathit{true} \;\vee\; \alpha_s(\mathit{enc}(q = \bot)_k) = \mathit{true}, & \text{(Deduction)} \\
 & \alpha_s(\mathit{enc}(r = \mathit{false})_k) = \mathit{true} \;\vee\; \alpha_s(\mathit{enc}(r = \bot)_k) = \mathit{true}, & \text{(Deduction)} \\
 & \alpha_{s'}(\mathit{enc}(p_j = \bot)_{k+1}) = \mathit{true} & \text{(Deduction)} \\
\Rightarrow & \alpha_s(\neg q[u]_k \wedge \neg q[t]_k) = \mathit{true} \;\vee\; \alpha_s(q[u]_k) = \mathit{true}, & \text{(Definition 8)} \\
 & \alpha_s(\neg r[u]_k \wedge \neg r[t]_k) = \mathit{true} \;\vee\; \alpha_s(r[u]_k) = \mathit{true}, & \text{(Definition 8)} \\
 & \alpha_{s'}(p_j[u]_{k+1}) = \mathit{true} & \text{(Definition 8)} \\
\Rightarrow & (\alpha_s(q[u]_k) = \mathit{false} \wedge \alpha_s(q[t]_k) = \mathit{false}) \;\vee\; \alpha_s(q[u]_k) = \mathit{true}, & \text{(Equivalent transformation)} \\
 & (\alpha_s(r[u]_k) = \mathit{false} \wedge \alpha_s(r[t]_k) = \mathit{false}) \;\vee\; \alpha_s(r[u]_k) = \mathit{true}, & \text{(Equivalent transformation)} \\
 & \alpha_{s'}(p_j[u]_{k+1}) = \mathit{true} & \\
\Rightarrow & \alpha_s(q[t]_k) = \mathit{false} \;\vee\; \alpha_s(q[u]_k) = \mathit{true}, & \text{(Equivalent transformation)} \\
 & \alpha_s(r[t]_k) = \mathit{false} \;\vee\; \alpha_s(r[u]_k) = \mathit{true}, & \text{(Equivalent transformation)} \\
 & \alpha_{s'}(p_j[u]_{k+1}) = \mathit{true} & \\
\Rightarrow & \alpha_s((q[u]_k \vee \neg q[t]_k) \wedge (r[u]_k \vee \neg r[t]_k)) = \mathit{true}, & \text{(Equivalent transformation)} \\
 & \alpha_{s'}(p_j[u]_{k+1}) = \mathit{true} &
\end{array}
$$

Hence, we have proven that $s(\mathit{choice}(q,r)) = \bot \;\wedge\; s'(p_j) = \bot$ implies the above fact about α_s and $\alpha_{s'}$, which we denote as *Fact 4*. By making use of Fact 4 we now can prove that $s(\mathit{choice}(q,r)) = \bot \;\wedge\; s'(p_j) = \bot$ also implies that $(\alpha_s(\mathit{enc}(q)_k) \wedge \alpha_{s'}(\mathit{enc}(p_j = \mathit{true})_{k+1})) \vee (\alpha_s(\mathit{enc}(r)_k) \wedge \alpha_{s'}(\mathit{enc}(p_j = \mathit{false})_{k+1})) \vee (\alpha_s(\mathit{enc}(\neg p \wedge$

$\neg q)_k[\bot/true]) \wedge \alpha_{s'}(enc(p_j = \bot)_{k+1}))$ is equivalent to *true*. For this, we transform this expression as follows:

$$
\begin{aligned}
& \quad (\alpha_s(enc(q)_k) \wedge \alpha_{s'}(enc(p_j = true)_{k+1})) \\
& \vee (\alpha_s(enc(r)_k) \wedge \alpha_{s'}(enc(p_j = false)_{k+1})) \\
& \vee (\alpha_s(enc(\neg q \wedge \neg r)_k[\bot/true]) \wedge \alpha_{s'}(enc(p_j = \bot)_{k+1})) \\
\equiv & \quad (\alpha_s((q[u]_k \wedge \bot) \vee (\neg q[u]_k \wedge q[t]_k)) \wedge \alpha_{s'}(\neg p_j[u]_{k+1} \wedge p_j[t]_{k+1})) \\
& \vee (\alpha_s((r[u]_k \wedge \bot) \vee (\neg r[u]_k \wedge r[t]_k)) \wedge \alpha_{s'}(\neg p_j[u]_{k+1} \wedge \neg p_j[t]_{k+1})) \\
& \vee (\alpha_s(((q[u]_k \wedge \bot) \vee (\neg q[u]_k \wedge \neg q[t]_k)) \wedge ((r[u]_k \wedge \bot) \vee (\neg r[u]_k \wedge \neg r[t]_k))[\bot/true]) \wedge \alpha_{s'}(p_j[u]_{k+1})) \\
& \quad \text{(Definition 9)} \\
\equiv & \quad (\alpha_s((q[u]_k \wedge \bot) \vee (\neg q[u]_k \wedge q[t]_k)) \wedge \alpha_{s'}(\neg p_j[u]_{k+1} \wedge p_j[t]_{k+1})) \\
& \vee (\alpha_s((r[u]_k \wedge \bot) \vee (\neg r[u]_k \wedge r[t]_k)) \wedge \alpha_{s'}(\neg p_j[u]_{k+1} \wedge \neg p_j[t]_{k+1})) \\
& \vee (\alpha_s((q[u]_k \vee (\neg q[u]_k \wedge \neg q[t]_k)) \wedge (r[u]_k \vee (\neg r[u]_k \wedge \neg r[t]_k))) \wedge \alpha_{s'}(p_j[u]_{k+1})) \\
& \quad \text{(Application of the substitution)} \\
\equiv & \quad (\alpha_s((q[u]_k \wedge \bot) \vee (\neg q[u]_k \wedge q[t]_k)) \wedge \alpha_{s'}(\neg p_j[u]_{k+1} \wedge p_j[t]_{k+1})) \\
& \vee (\alpha_s((r[u]_k \wedge \bot) \vee (\neg r[u]_k \wedge r[t]_k)) \wedge \alpha_{s'}(\neg p_j[u]_{k+1} \wedge \neg p_j[t]_{k+1})) \\
& \vee (\alpha_s((q[u]_k \vee \neg q[t]_k) \wedge (r[u]_k \vee \neg r[t]_k)) \wedge \alpha_{s'}(p_j[u]_{k+1})) \\
& \quad \text{(Equivalent transformation)} \\
\equiv & \quad (\alpha_s((q[u]_k \wedge \bot) \vee (\neg q[u]_k \wedge q[t]_k)) \wedge false) \\
& \vee (\alpha_s((r[u]_k \wedge \bot) \vee (\neg r[u]_k \wedge r[t]_k)) \wedge false) \\
& \vee (true \wedge true) \\
& \quad \text{(Fact 4)} \\
\equiv & \quad true \\
& \quad \text{(Equivalent transformation)}
\end{aligned}
$$

This completes the '⇒'-direction of the proof. Next we prove the '⇐'-direction.

$$
\begin{aligned}
& \quad (\alpha_s(enc(q)_k) \wedge \alpha_{s'}(enc(p_j = true)_{k+1})) \\
& \vee (\alpha_s(enc(r)_k) \wedge \alpha_{s'}(enc(p_j = false)_{k+1})) \\
& \vee (\alpha_s(enc(\neg q \wedge \neg r)_k[\bot/true]) \wedge \alpha_{s'}(enc(p_j = \bot)_{k+1})) \\
& = true \\
\equiv & \quad (\alpha_s((q[u]_k \wedge \bot) \vee (\neg q[u]_k \wedge q[t]_k)) \wedge \alpha_{s'}(\neg p_j[u]_{k+1} \wedge p_j[t]_{k+1})) \\
& \vee (\alpha_s((r[u]_k \wedge \bot) \vee (\neg r[u]_k \wedge r[t]_k)) \wedge \alpha_{s'}(\neg p_j[u]_{k+1} \wedge \neg p_j[t]_{k+1})) \\
& \vee (\alpha_s((q[u]_k \vee \neg q[t]_k) \wedge (r[u]_k \vee \neg r[t]_k)) \wedge \alpha_{s'}(p_j[u]_{k+1})) \\
& = true \\
& \quad \text{(Compare transformations for '}\Rightarrow\text{'-direction)} \\
\Rightarrow & \quad (\alpha_s(q[u]_k) = false \wedge \alpha_s(q[t]_k) = true \wedge \alpha_{s'}(p_j[u]_{k+1}) = false \wedge \alpha_{s'}(p_j[t]_{k+1}) = true) && \text{(I)} \\
& \vee (\alpha_s(r[u]_k) = false \wedge \alpha_s(r[t]_k) = true \wedge \alpha_{s'}(p_j[u]_{k+1}) = false \wedge \alpha_{s'}(p_j[t]_{k+1}) = false) && \text{(II)} \\
& \vee (\alpha_s(q[u]_k) = true \wedge \alpha_s(r[u]_k) = true \wedge \alpha_{s'}(p_j[u]_{k+1}) = true) && \text{(III)} \\
& \vee (\alpha_s(\neg q[t]_k) = false \wedge \alpha_s(r[u]_k) = true \wedge \alpha_{s'}(p_j[u]_{k+1}) = true) && \text{(IV)} \\
& \vee (\alpha_s(q[u]_k) = true \wedge \alpha_s(r[t]_k) = false \wedge \alpha_{s'}(p_j[u]_{k+1}) = true) && \text{(V)} \\
& \vee (\alpha_s(\neg q[t]_k) = false \wedge \alpha_s(r[t]_k) = false \wedge \alpha_{s'}(p_j[u]_{k+1}) = true) && \text{(VI)} \\
& \quad \text{(Deduction)}
\end{aligned}
$$

Hence, we have shown that if $(\alpha_s(enc(q)_k) \wedge \alpha_{s'}(enc(p_j = true)_{k+1})) \vee (\alpha_s(enc(r)_k) \wedge \alpha_{s'}(enc(p_j = false)_{k+1})) \vee (\alpha_s(enc(\neg p \wedge \neg q)_k[\bot/true]) \wedge \alpha_{s'}(enc(p_j = \bot)_{k+1}))$ is equivalent to *true* then the definition of the assignments α_s and $\alpha_{s'}$ must be conform with one of the above lines (I) to (VI). We now show that if we take any of these lines as a constraint with regard to α_s and $\alpha_{s'}$ then for the corresponding states s and s' the equation $s'(p_j) = s(choice(q, r))$ yields *true* as well.

Under Constraint (I):

For the left side of the equation we get:

$$
\begin{aligned}
& \alpha_{s'}(enc(p_j = s'(p_j))_{k+1}) = true \\
& \text{(Deduction from Definition 13)} \\
& \alpha_{s'}(\neg p_j[u]_{k+1} \wedge p_j[t]_{k+1}) = true \\
& \text{(Deduction from Constraint (I))} \\
\Rightarrow\ & enc(p_j = s'(p_j))_{k+1} \ = \ \neg p_j[u]_{k+1} \wedge p_j[t]_{k+1} \\
& \text{(Definition 8)} \\
\Rightarrow\ & s'(p_j) = true \\
& \text{(Definition 8)}
\end{aligned}
$$

For the right side of the equation we get:

$$
\begin{aligned}
& \alpha_s(enc(q = s(q))_k) = true \\
& \text{(Deduction from Definition 13)} \\
& \alpha_s(\neg q[u]_k \wedge q[t]_k) = true \\
& \text{(Deduction from Constraint (I))} \\
\Rightarrow\ & enc(q = s(q))_k \ = \ \neg q[u]_k \wedge q[t]_k \\
& \text{(Definition 8)} \\
\Rightarrow\ & s(q) = true \\
& \text{(Definition 8)} \\
\Rightarrow\ & s(choice(q, r)) = true \\
& \text{(Definition 9)}
\end{aligned}
$$

Hence, the equation $s'(p_j) = s(choice(q, r))$ yields *true* under Constraint (I). The proofs under the other constraints are analogous. (Note that the abstraction technique that we apply guarantees that for an expression $choice(a, b)$ the expressions a and b are never *true* at the same time. Hence, $s(a) = true$ allows us to conclude that $s(b)$ is not *true* and vice versa.) This completes the proof of the '$\Leftarrow$'-direction. The proofs of the remaining cases from the table in Figure 2 are analogous.

(C) $L(\langle l, s\rangle, p_j) \ \equiv \ \alpha_{\langle l,s\rangle}(enc(p_j)_k)$ can be proven by showing that the following implications hold:

1. $L(\langle l, s\rangle, p_j) = true \ \Rightarrow \ \alpha_{\langle l,s\rangle}(enc(p_j)_k) = true$

2. $L(\langle l, s\rangle, p_j) = false \ \Rightarrow \ \alpha_{\langle l,s\rangle}(enc(p_j)_k) = false$

3. $L(\langle l, s\rangle, p_j) = \perp \ \Rightarrow \ \alpha_{\langle l,s\rangle}(enc(p_j)_k) = \perp$

We prove the first case. The proofs of the remaining cases are analogous.

$L(\langle l,s\rangle, p_j) = \mathit{true}$
(Premise)
$\alpha_s(enc(s)_k) = \mathit{true}$
(Definition 13)

$\Rightarrow s(p_j) = \mathit{true}$
(Definition 7 in [1])
$\alpha_s(enc(p_j = s(p_j))_k) = \mathit{true}$
(Definition 8)

$\Rightarrow \alpha_s(enc(p_j = \mathit{true})_k) = \mathit{true}$
(Deduction)

$\Rightarrow \alpha_s(\neg p_j[u]_k \wedge p_j[t]_k) = \mathit{true}$
(Definition 8)

$\Rightarrow \alpha_s(p_j[u]_k) = \mathit{false}, \alpha_s(p_j[t]_k) = \mathit{true}$
(Deduction)

$\Rightarrow \alpha_s((p_j[u]_k \wedge \bot) \vee (\neg p_j[u]_k \wedge p_j[t]_k)) = \mathit{true}$
(Deduction)

$\Rightarrow \alpha_s(enc(p_j)_k) = \mathit{true}$
(Definition 9)

(D) $L(\langle l,s\rangle, (loc_i = \hat{l}_i)) \;\equiv\; \alpha_{\langle l,s\rangle}(enc(\hat{l}_i)_k)$ can be proven by showing that the following implications hold:

1. $L(\langle l,s\rangle, (loc_i = \hat{l}_i)) = \mathit{true} \;\Rightarrow\; \alpha_{\langle l,s\rangle}(enc(\hat{l}_i)_k) = \mathit{true}$
2. $L(\langle l,s\rangle, (loc_i = \hat{l}_i)) = \mathit{false} \;\Rightarrow\; \alpha_{\langle l,s\rangle}(enc(\hat{l}_i)_k) = \mathit{false}$

We prove the first case. The proofs of the remaining case are analogous.

$L(\langle l,s\rangle, (loc_i = \hat{l}_i)) = \mathit{true}$
(Premise)
$\alpha_l(enc(l)_k) = \mathit{true}$
(Definition 13)

$\Rightarrow l_i = \hat{l}_i$
(Definition 7 in [1])
$\alpha_l(\bigwedge_{i=1}^{n} enc(l_i)_k) = \mathit{true}$
(Definition 7)

$\Rightarrow \alpha_s(enc(\hat{l}_i)_k) = \mathit{true}$
(Deduction)

This completes the proof of Lemma 1.
□

We now have that for each b-bounded path $\pi = \langle l^0, s^0\rangle, \ldots, \langle l^b, s^b\rangle$, in a Kripke structure M, corresponding to an abstracted system Sys, the assignment α_π characterises a b-bounded path in the encoding $[\![Sys]\!]_b$ with the same transition and labelling values as π, and vice versa. For the correctness of Theorem 1 we still have to show that the evaluation of an LTL property ψ on ψ yields the same result as $\alpha_\pi([\![\psi]\!]_b)$:

Lemma 2. *Let π be a b-bounded path and α_π the assignment characterising π. Moreover, let ψ be an LTL formula and $[\![\psi]\!]_b$ its encoding. Then*

$$[\pi \models_b^k \psi] \;\equiv\; \alpha_\pi([\![\psi]\!]_b)$$

Proof of Lemma 2.
Induction on the structure of ψ. We only present some cases. The remaining ones are proven analogously.

$$[\pi \models_b^k p_j] = L(\pi^k, p_j) = \alpha_\pi(enc(p_j)_k) = \alpha_\pi([\![\psi]\!]_b)$$

(Definition 6, Lemma 1, Definition 11)

$$[\pi \models_b^k \mathbf{F}\psi] = \bigvee_{j=k}^{b}[\pi \models_b^j \psi] = \bigvee_{j=k}^{b} \alpha_\pi([\![\psi]\!]_b^j) = \alpha_\pi(\bigvee_{j=k}^{b}[\![\psi]\!]_b^j) = \alpha_\pi([\![\mathbf{F}\psi]\!]_b^k)$$

(Definition 6, Induction, Definition 11)
□

The correctness of Theorem 1 now follows from Lemma 1 and Lemma 2 together.
□

References

1. Schrieb, J., Wehrheim, H., Wonisch, D.: Three-valued spotlight abstractions. In: Cavalcanti, A., Dams, D.R. (eds.) FM 2009: Formal Methods, LNCS, vol. 5850, pp. 106–122. Springer-Verlag Berlin Heidelberg (2009)