MGHyper: Checking Satisfiability of HyperLTL formulas beyond the $\exists^*\forall^*$ Fragment*

Bernd Finkbeiner, Christopher Hahn, and Tobias Hans

Reactive Systems Group
Saarland University
lastname@react.uni-saarland.de

Abstract. Hyperproperties are properties that refer to multiple computation traces. This includes many information-flow security policies, such as observational determinism, (generalized) noninterference, and noninference, and other system properties like symmetry or Hamming distances between in error-resistant codes. We introduce MGHyper, a tool for automatic satisfiability checking and model generation for hyperproperties expressed in HyperLTL. Unlike previous satisfiability checkers, MGHyper is not limited to the decidable $\exists^*\forall^*$ fragment of HyperLTL, but provides a semi-decisionprocedure for the full logic. An important application of MGHyper is to automatically check equivalences between different hyperproperties (and different formalizations of the same hyperproperty) and to build counterexamples that disprove a certain claimed implication. We describe the semi-decisionprocedure implemented in MGHyper and report on experimental results obtained both with typical hyperproperties from the literature and with randomly generated HyperLTL formulas.

1 Introduction

HyperLTL [3] extends linear-time temporal logic (LTL) [20] with explicit quantification over traces. This makes it possible to express hyperproperties [4] like noninterference [13] or symmetry [11], which refer to multiple traces at the same time. Such properties are not expressible in LTL, or even in the branching-time temporal logics CTL [2] and CTL* [6]. For example, *noninference* [18] is a variant of noninterference stating that, for all system traces, the low-observable behavior must not change when all high inputs are replaced by a dummy input. The following HyperLTL formula expresses this policy: $\forall\pi.\ \exists\pi'.\ (\mathsf{G}\lambda_{\pi'}) \wedge \pi =_{L,out} \pi'$. HyperLTL is supported by model checking [11] and runtime monitoring tools [9, 10]. There is also a decision procedure, EAHyper [8], which checks the satisfiability of a given formula from the $\exists^*\forall^*$ fragment of HyperLTL. EAHyper is based on a reduction from HyperLTL satisfiability to LTL satisfiability [7]. A major application of EAHyper is to check equivalences between alternation-free HyperLTL formulas, i.e., formulas that either contain only universal quantifiers or only existential quantifers. Such equivalences can be expressed in the $\exists^*\forall^*$ fragment. It is

* This work was partially supported by the German Research Foundation (DFG) in the Collaborative Research Center 1223 and by the European Research Council (ERC) Grant OSARES (No. 683300).

impossible, however, to handle formulas that contain a $\forall\exists$ quantifier alternation, as, for example, in noninference. This is unfortunate, because such a quantifier alternation is often needed, in particular to account for nondeterminism. A popular example is *generalized noninterference* [14]: $\forall\pi.\forall\pi'.\exists\pi''.\ \pi =_{H,\mathsf{in}} \pi'' \wedge \pi' =_{L,out} \pi''$. The formula expresses that for every possible high-security input (seen on some trace π) and every possible low-security observations (seen on some trace π') there exists a nondeterministic execution π'' where the high-security input and the low-security observations happen together. Hence, the observer cannot conclude, after making the low-security observations, that any specific high-security input actually occurred. Other properties that need a $\forall\exists$ quantifier alternation include restrictiveness [15], separability [17], and forward correctability [19]. For formulas outside the $\exists^*\forall^*$ fragment, it is no longer possible to reduce the HyperLTL satisfiability problem to the LTL satsifiability problem: the HyperLTL satisfiability problem is, in fact, undecidable [7]. In this paper, we present the first semi-decisionprocedure for full HyperLTL. Our approach is based on a reduction to quantified boolean formulas (QBF) [12] and has been implemented in the MGHyper tool. MGHyper can be used to analyze and develop hyperproperties and, especially, generate models that disprove equivalences or implications between different hyperproperties or different formalizations of the same hyperproperty. For example, comparing noninference to generalized noninterference, MGHyper instantly demonstrates that the two properties are *not* equivalent.

2 A Semi-Decision Procedure for HyperLTL-SAT

A *hyperproperty* is a *set of sets of traces*. Hyperproperties can be expressed in HyperLTL, which generalizes LTL with explicit trace quantification:

$$\begin{aligned} \psi &::= \exists\pi.\ \psi \mid \forall\pi.\ \psi \mid \varphi \\ \varphi &::= a_\pi \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \mid \mathit{true} \end{aligned}$$

where Q is an existential or universal quantifier, $a \in AP$ is an atomic proposition and $\pi \in \mathcal{V}$ is a trace variable of an infinitely supply $\mathcal{V}$. Logical connectives and the temporal operators F, G, W and R are defined as in LTL. The semantics of HyperLTL is defined as follows.

$\Pi \models_T$	$\exists\pi.\psi$	iff	there exists $t \in T$: $\Pi[\pi \mapsto t] \models_T \psi$
$\Pi \models_T$	$\forall\pi.\psi$	iff	for all $t \in T$: $\Pi[\pi \mapsto t] \models_T \psi$
$\Pi \models_T$	a_π	iff	$a \in \Pi(\pi)[0]$
$\Pi \models_T$	$\neg\psi$	iff	$\Pi \not\models_T \psi$
$\Pi \models_T$	$\psi_1 \vee \psi_2$	iff	$\Pi \models_T \psi_1$ or $\Pi \models_T \psi_2$
$\Pi \models_T$	$\mathsf{X}\psi$	iff	$\Pi[1,\infty] \models_T \psi$
$\Pi \models_T$	$\psi_1\,\mathsf{U}\,\psi_2$	iff	there exists $i \geq 0 : \Pi[i,\infty] \models_T \psi_2$ and for all $0 \leq j < i$ we have $\Pi[j,\infty] \models_T \psi_1$

,where $\Pi : \mathcal{V} \mapsto TR$ is the trace assignment function, which maps trace variables to traces, denoted by $\Pi[\pi \mapsto t]$. The suffixes of all traces π starting at step i

is denoted by $\Pi[i, \infty]$. *HyperLTL-SAT* is the problem to decide, if a non-empty trace set T exists, such that $\{\} \models_T \psi$.

MGHyper takes an *arbitrary* HyperLTL formula of the following form as input: $Q_0 \boldsymbol{\pi_0} \dots Q_n \boldsymbol{\pi_n}.\varphi$, where $Q_i \in \{\exists, \forall\}$ and $\boldsymbol{\pi_0} \dots \boldsymbol{\pi_n}$ are vectors over $\mathcal{V}$. MGHyper evaluates to "sat" if and only if the formula is satisfiable. Basically, MGHyper checks whether or not there exists a trace set of size m that satisfies the HyperLTL formula under consideration. The procedure starts with trace sets of size 1, and increment m until a witness is found, leading to the following theorem.

Theorem 1. *HyperLTL-SAT is RE-complete.*

Proof. We prove membership by constructing a QBF formula φ^m_{QBF}, which is satisfiable if the given HyperLTL formula φ is satisfiable by a trace set of size m. The basic idea of the encoding of a HyperLTL formula to a QBF formula is threefold: 1) we construct a quantifier prefix that resembles the quantifier structure in the given HyperLTL formula, 2) we construct a premise that links trace variables to actual traces, and 3) we unroll the LTL suffix into a SAT-encoding. The third step follows the unrolling presented in [1] and will, due to space reasons, not be discussed. We refer to the maximum trace-unrolling bound as k (not to confuse with m), which is exponential in the size of the LTL suffix.

1) Prefix. Let S be a set and k a natural number, we define Traces^k_S as $\{a^i_s \mid 0 \leq i < k, \forall s \in S, \forall a \in AP\}$, which we use as the trace representation inside the QBF encoding. Let $\varphi := Q_0 \boldsymbol{\pi_0} \dots Q_n \boldsymbol{\pi_n}.\psi$ be a HyperLTL formula. The quantifier prefix of the resulting QBF introduces existential quantifiers, representing the trace set T of size m (the witness for satisfiability). The trace variables are quantified according to their quantifier in the HyperLTL formula:

$$\mathit{Prefix}(\varphi) = \exists \mathit{Traces}^k_T.Q_0\, \mathit{Traces}^k_{\boldsymbol{\pi_0}}.Q_1\, \mathit{Traces}^k_{\boldsymbol{\pi_1}}. \dots .Q_n\, \mathit{Traces}^k_{\boldsymbol{\pi_n}}.$$

2) Linking. We construct a premise to link trace variables to the trace witnesses in Traces^k_T. For every quantifier Q_i a subpremise P_{Q_i} is constructed first, which represents the mapping of all trace variables in $\boldsymbol{\pi_i}$. Mapping each trace variable to traces reassembles the trace assignment function from the HyperLTL semantics and is encoded by ensuring that the boolean variables with the same atomic proposition in the same step share the same truth value.

$$P_{Q_i} := \left[\bigwedge_{\pi \in \boldsymbol{\pi_i}} \bigvee_{t_i \in T} \left[\bigwedge_{\substack{(a^j_{t_i}, a^j_\pi) \in \\ \mathit{Traces}^k_{t_i} \times \mathit{Traces}^k_\pi}} a^j_{t_i} \leftrightarrow a^j_\pi \right] \right] \tag{1}$$

The linking mechanism is a combination of the subpremises. The boolean connective between the different supremises depends on the corresponding quantifier:

$$\mathit{Linking}(\varphi) := P^k_{Q_0} \circ_{Q_0} P^k_{Q_1} \circ_{Q_1} \dots P^k_{Q_{n-1}} \circ_{Q_{n-1}} P^k_{Q_n} \circ_{Q_n}, \tag{2}$$

where $\circ_{Q_i}$ equals $\rightarrow$, if $Q_i = \forall$, and $\circ_{Q_i}$ equals $\wedge$ if $Q_i = \exists$. Together with *3) the unrolling* of the LTL suffix [1], the constructed φ^m_{QBF} a QBF formula

is satisfiable if the HyperLTL formula φ is satisfiable. Hardness follows from a reduction from Post's Correspondence Problem [7]. □

Example 1. Consider the HyperLTL formula $\varphi := \forall\pi_0\exists\pi_1\exists\pi_2.(a_{\pi_0} \wedge (a_{\pi_1} \rightarrow \neg b_{\pi_1} \wedge a_{\pi_2} \rightarrow b_{\pi_2}))$. Note that, for the sake of simplicity, the example LTL formula does not contain temporal operators. In the first iteration, MGHyper tries to guess a trace set T of size 1 and will not find a satisfying assignment for the constructed QBF formula. In the second iteration, though, MGHyper constructs the following QBF formula, with $T_2 = \{\{a^0_{t_0}, b^0_{t_0}\}, \{a^0_{t_1}, b^0_{t_1}\}\}$.

$$\exists \mathit{Traces}^0_{T_2} \forall \mathit{Traces}^0_{\pi_0} \exists \mathit{Traces}^0_{\pi_1} \exists \mathit{Traces}^0_{\pi_2}. \begin{bmatrix} \vee (a^0_{\pi_0} \leftrightarrow a^0_{t_0} \wedge b^0_{\pi_0} \leftrightarrow b^0_{t_0}) \\ (a^0_{\pi_0} \leftrightarrow a^0_{t_1} \wedge b^0_{\pi_0} \leftrightarrow b^0_{t_1}) \end{bmatrix}$$
$$\rightarrow \left(\begin{bmatrix} \begin{pmatrix} \vee & [a^0_{\pi_1} \leftrightarrow a^0_{t_0} \wedge b^0_{\pi_1} \leftrightarrow b^0_{t_0}] \\ & [a^0_{\pi_1} \leftrightarrow a^0_{t_1} \wedge b^0_{\pi_1} \leftrightarrow b^0_{t_1}] \end{pmatrix} \\ \wedge \\ \begin{pmatrix} \vee & [a^0_{\pi_2} \leftrightarrow a^0_{t_0} \wedge b^0_{\pi_2} \leftrightarrow b^0_{t_0}] \\ & [a^0_{\pi_2} \leftrightarrow a^0_{t_1} \wedge b^0_{\pi_2} \leftrightarrow b^0_{t_1}] \end{pmatrix} \end{bmatrix} \wedge (a^0_{\pi_0} \wedge (a^0_{\pi_1} \rightarrow \neg b^0_{\pi_1} \wedge a^0_{\pi_2} \rightarrow b^0_{\pi_2})) \right)$$

This QBF formula is satisfied by the assignment $\mathcal{A} = \{a^0_{t_0}, b^0_{t_0}, a^0_{t_1}, \neg b^0_{t_1}\}$ for the existentially quantified variables, which represent the traces (of length one in this example). There are four possible assignment for the universally quantified boolean variables. For $\{a^0_{\pi_0}, b^0_{\pi_0}\}$ or $\{a^0_{\pi_0}, \neg b^0_{\pi_0}\}$ we can map the existentially quantified traces variables $\pi_1 \mapsto t_1$ and $\pi_2 \mapsto t_0$, which add $\{a^0_{\pi_1}, \neg b^0_{\pi_1}, a^0_{\pi_2}, b^0_{\pi_2}\}$ to $\mathcal{A}$, such that $\mathcal{A}$ satisfies the formula. In the other two cases,$\{\neg a^0_{\pi_0}, \neg b^0_{\pi_0}\}$ or $\{\neg a^0_{\pi_0}, b^0_{\pi_0}\}$, we cannot map to $\neg a^0_{\pi_0}$, which leads to a false evaluation of $P^0_{Q_0}$ and therefore to a true evaluation of the formula. From $\mathcal{A}$ we can now follow that $\{\{a, b\}^\omega, \{a\}^\omega\}$ is a model that satisfies φ.

3 Experimental Results

MGHyper is implemented in Ocaml and supports UNIX-based operation systems. We tested our tool against different benchmarks on a virtual machinerunning an Ubuntu (64-Bit) 14.04LTS installation on an Intel Core i7-4710MQ with 2,50GH on 4 kernels and 8GB RAM.

Counter Examples for Implication of Security Polices. A main application of MGHyper is to check if two arbitrary HyperLTL formulas do not imply each other: we check if the negation of the implication is satisfiable. The first benchmark checks the implication of *General Noninterference* [4] ((GNI):$\forall\pi_1.\forall\pi_2.\exists\pi_3\mathsf{G}(I^{high}_{\pi_1} = I^{high}_{\pi_3}) \wedge \mathsf{G}(O^{low}_{\pi_2} = O^{low}_{\pi_3})$), *Noninerference* ((NI): $\forall\pi_1.\ \exists\pi_2.\ (\mathsf{G}\lambda_{\pi_2}) \wedge \mathsf{G}(O_{\pi_1} = O_{\pi_2})$) and several formalizations of *Observational Determinsim* [16, 21, 22]: (OD): $\forall\pi_1.\forall\pi_2.(I^{low}_{\pi_1} = I^{low}_{\pi_2}) \rightarrow \mathsf{G}(O^{low}_{\pi_1} = O^{low}_{\pi_2})$, (GOD): $\forall\pi_1.\forall\pi_2.\mathsf{G}(I^{low}_{\pi_1} = I^{low}_{\pi_2}) \rightarrow \mathsf{G}(O^{low}_{\pi_1} = O^{low}_{\pi_2})$, and (WOD):$\forall\pi_1.\forall\pi_2.(I^{low}_{\pi_1} = I^{low}_{\pi_2})\,\mathsf{W}\,(O^{low}_{\pi_1} \neq O^{low}_{\pi_2})$. MGHyper shows that none of the formalizations of observational determinism does imply general non-interference or noninfernce. Furthermore, it shows that generalized noninterference does not imply noninference. Every check was done in under 0.05 seconds.

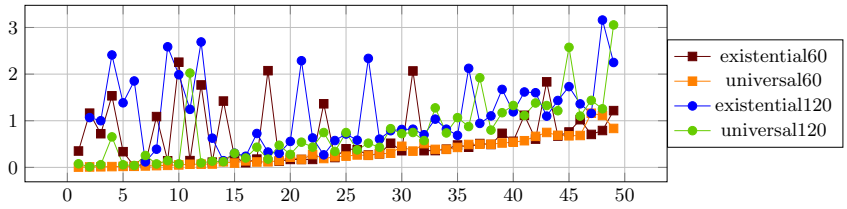

Fig. 1: Runtime for random HyperLTL formulas of size 60 and 120. A formula consists of up to 50 trace variables starting universally or existentially quantified.

Table 1: Random $\exists^n\forall^m$ formulas: instances solved (sol) in 120 s and average time (avgt) in s for 100 random formulas of size 60 with 15 atomic propositions.

avgt	sol	avgt	sol	avgt	sol	avgt	sol	avgt	sol	avgt	sol	avgt	sol	avgt	sol	avgt	sol	avgt	sol
$\exists^1\forall^1$		$\exists^1\forall^2$		$\exists^1\forall^3$		$\exists^1\forall^4$		$\exists^1\forall^5$		$\exists^1\forall^6$		$\exists^1\forall^7$		$\exists^1\forall^8$		$\exists^1\forall^9$		$\exists^1\forall^{10}$	
0.472	96	0.599	95	1.371	96	1.537	96	1.322	96	1.587	96	0.529	92	2.823	96	3.166	94	2.303	97
$\exists^2\forall^1$		$\exists^2\forall^2$		$\exists^2\forall^3$		$\exists^2\forall^4$		$\exists^2\forall^5$		$\exists^2\forall^6$		$\exists^2\forall^7$		$\exists^2\forall^8$		$\exists^2\forall^9$		$\exists^2\forall^{10}$	
6.772	80	2.743	85	2.698	93	6.319	94	4.233	97	4.107	92	3.162	87	2.906	94	4.847	92	2.131	96
$\exists^3\forall^1$		$\exists^3\forall^2$		$\exists^3\forall^3$		$\exists^3\forall^4$		$\exists^3\forall^5$		$\exists^3\forall^6$		$\exists^3\forall^7$		$\exists^3\forall^8$		$\exists^3\forall^9$		$\exists^3\forall^{10}$	
5.533	79	9.921	81	5.836	82	4.851	88	7.589	82	3.852	82	9.975	82	5.222	79	6.328	77	5.044	83
$\exists^4\forall^1$		$\exists^4\forall^2$		$\exists^4\forall^3$		$\exists^4\forall^4$		$\exists^4\forall^5$		$\exists^4\forall^6$		$\exists^4\forall^7$		$\exists^4\forall^8$		$\exists^4\forall^9$		$\exists^4\forall^{10}$	
10.316	75	8.669	80	5.631	83	3.722	83	5.843	73	5.62	81	10.074	79	6.955	76	8.037	85	5.938	79
$\exists^5\forall^1$		$\exists^5\forall^2$		$\exists^5\forall^3$		$\exists^5\forall^4$		$\exists^5\forall^5$		$\exists^5\forall^6$		$\exists^5\forall^7$		$\exists^5\forall^8$		$\exists^5\forall^9$		$\exists^5\forall^{10}$	
5.431	71	5.009	80	2.812	69	8.514	81	4.501	76	6.255	83	1.574	76	3.616	76	5.85	79	7.486	80
$\exists^6\forall^1$		$\exists^6\forall^2$		$\exists^6\forall^3$		$\exists^6\forall^4$		$\exists^6\forall^5$		$\exists^6\forall^6$		$\exists^6\forall^7$		$\exists^6\forall^8$		$\exists^6\forall^9$		$\exists^6\forall^{10}$	
3.53	78	4.378	74	3.503	71	3.057	76	4.354	71	4.513	81	3.492	79	4.836	79	6.289	80	6.0	74
$\exists^7\forall^1$		$\exists^7\forall^2$		$\exists^7\forall^3$		$\exists^7\forall^4$		$\exists^7\forall^5$		$\exists^7\forall^6$		$\exists^7\forall^7$		$\exists^7\forall^8$		$\exists^7\forall^9$		$\exists^7\forall^{10}$	
4.33	74	3.173	70	1.789	72	7.187	69	4.99	78	5.584	74	4.783	77	7.558	75	7.744	74	6.043	78
$\exists^8\forall^1$		$\exists^8\forall^2$		$\exists^8\forall^3$		$\exists^8\forall^4$		$\exists^8\forall^5$		$\exists^8\forall^6$		$\exists^8\forall^7$		$\exists^8\forall^8$		$\exists^8\forall^9$		$\exists^8\forall^{10}$	
5.681	81	4.617	79	6.803	74	5.563	75	6.219	80	5.999	74	3.013	73	2.041	72	6.146	75	3.997	75
$\exists^9\forall^1$		$\exists^9\forall^2$		$\exists^9\forall^3$		$\exists^9\forall^4$		$\exists^9\forall^5$		$\exists^9\forall^6$		$\exists^9\forall^7$		$\exists^9\forall^8$		$\exists^9\forall^9$		$\exists^9\forall^{10}$	
3.509	77	2.514	72	5.659	68	1.345	78	4.379	72	3.914	73	3.422	71	1.784	66	6.903	72	5.142	77
$\exists^{10}\forall^1$		$\exists^{10}\forall^2$		$\exists^{10}\forall^3$		$\exists^{10}\forall^4$		$\exists^{10}\forall^5$		$\exists^{10}\forall^6$		$\exists^{10}\forall^7$		$\exists^{10}\forall^8$		$\exists^{10}\forall^9$		$\exists^{10}\forall^{10}$	
3.986	81	4.553	70	5.777	70	4.791	75	8.284	75	1.534	72	4.338	71	4.18	76	5.512	65	4.529	75

Random Formulas. We tested MGHyper against different benchmarks of random formulas. *Quantfier Alternation:* We created HyperLTL formulas with up to 49 quantifier alternations and 15 atomic propostion using randltl [5]. For each number of alternations we tested 100 formulas of size 60 and 120, where 50 start with $\exists$ and 50 with $\forall$. The size is the size argument provided for randltl [5]. The runtimes are shown in Figure 1. *Quantifier Ordering: $\exists^n\forall^m$ Formulas:* For the sake of comparing MGHyper with EAHyper, we tested MGHyper on the largest decidable fragment of HyperLTL, which is the $\exists^*\forall^*$-fragment. We scaled in the number of existential and universal quantifiers, showing that MGHyper is able to solve formulas with up to 10 existential and 10 universal quantifier. In comparison, EAHyper, implementing the first decision procedure for HyperLTL-SAT, already runs out of memory after 5 existential and 5 universal quantifiers.

4 Conclusion

We have presented MGHyper, the first semi-decisionprocedure for checking the satisfiability of HyperLTL formulas beyond the decidable $\exists^*\forall^*$ fragment. An application of MGHyper is the analysis and development of hyperproperties and, especially, the generation of models that disprove equivalences or implications between different hyperproperties. In comparison to the existing decision procedure EAHyper, MGHyper not only handles the much larger class of hyperproperties, it also outperforms, as our experiments show, EAHyper within the $\exists^*\forall^*$ fragment.

References

1. Biere, A., Cimatti, A., Clarke, E.M., Strichman, O., Zhu, Y.: Bounded model checking. Advances in computers (2003)
2. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching-time temporal logic. In: Logics of Programs, Workshop (1981)
3. Clarkson, M.R., Finkbeiner, B., Koleini, M., Micinski, K.K., Rabe, M.N., Sánchez, C.: Temporal logics for hyperproperties. POST (2014)
4. Clarkson, M.R., Schneider, F.B.: Hyperproperties. Journal of Comp. Sec. (2010)
5. Duret-Lutz, A.: Manipulating LTL formulas using spot 1.0. In: ATVA (2013)
6. Emerson, E.A., Halpern, J.Y.: "sometimes" and "not never" revisited: on branching versus linear time temporal logic. J. ACM (1986)
7. Finkbeiner, B., Hahn, C.: Deciding hyperproperties. In: CONCUR (2016)
8. Finkbeiner, B., Hahn, C., Stenger, M.: Eahyper: Satisfiability, implication, and equivalence checking of hyperproperties. In: CAV (2017)
9. Finkbeiner, B., Hahn, C., Stenger, M., Tentrup, L.: Monitoring hyperproperties. In: RV (2017)
10. Finkbeiner, B., Hahn, C., Stenger, M., Tentrup, L.: Rvhyper : A runtime verification tool for temporal hyperproperties. In: TACAS (2018)
11. Finkbeiner, B., Rabe, M.N., Sánchez, C.: Algorithms for model checking hyperltl and hyperctl ^*. In: CAV (2015)
12. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
13. Goguen, J.A., Meseguer, J.: Security policies and security models. In: S&P (1982)
14. McCullough, D.: Noninterference and the composability of security properties. In: S&P (1988)
15. McCullough, D.: A hookup theorem for multilevel security. IEEE Trans. Software Eng. (1990)
16. McLean, J.: Proving noninterference and functional correctness using traces. Journal of Computer Security 1(1), 37–58 (1992)
17. McLean, J.: A general theory of composition for trace sets closed under selective interleaving functions. In: S&P (1994)
18. McLean, J.: A general theory of composition for a class of "possibilistic" properties. IEEE Trans. Software Eng. (1996)
19. Millen, J.K.: Unwinding forward correctability. J. of Computer Security (1995)
20. Pnueli, A.: The temporal logic of programs. In: Foundations of Computer Science (1977)
21. Roscoe, A.W.: CSP and determinism in security modelling. In: S&P (1995)
22. Zdancewic, S., Myers, A.C.: Observational determinism for concurrent program security. In: CSFW-16