# Specifying and Verifying Secrecy in Workflows with Arbitrarily Many Agents

Bernd Finkbeiner[1], Helmut Seidl[2], and Christian Müller[2]

[1] Universität des Saarlandes, 66123 Saarbrücken, Germany
finkbeiner@cs.uni-saarland.de
[2] TU München, 80333 München, Germany
{seidl,christian.mueller}@in.tum.de

**Abstract.** Web-based workflow management systems, like EasyChair, HealthVault, Ebay, or Amazon, often deal with confidential information such as the identity of reviewers, health data, or credit card numbers. Because the number of participants in the workflow is in principle unbounded, it is difficult to describe the information flow policy of such systems in specification languages that are limited to a fixed number of agents. We introduce a first-order version of HyperLTL, which allows us to express information flow requirements in workflows with arbitrarily many agents. We present a bounded model checking technique that reduces the violation of the information flow policy to the satisfiability of a first-order formula. We furthermore identify conditions under which the resulting satisfiability problem is guaranteed to be decidable.

## 1 Introduction

Web-based workflow management systems allow diverse groups of users to collaborate efficiently on complex tasks. For example, conference management systems like EasyChair let authors, reviewers, and program committees collaborate on the organization of a scientific conference; health management systems like HealthVault let family members, doctors, and other health care providers collaborate on the management of a patient's care; shopping sites like Amazon or Ebay let merchants, customers, as well as various other agents responsible for payment, customer service, and shipping, collaborate on the purchase and delivery of products.

Since the information maintained in such systems is often confidential, the workflows must carefully manage who has access to what information in a particular stage of the workflow. For example, in a conference management system, PC members must declare conflicts of interest, and they should only see reviews of papers where no conflict exists. Authors eventually get access to reviews of their papers, but only when the process has reached the official notification stage, and without identifying information about the reviewers.

It is difficult to characterize the legitimate information flow in such systems with standard notions of secrecy. Classic information flow policies are often too strong. For example, *noninterference* [12] requires that the PC member cannot observe any difference when classified input, such as the reviews of papers where the PC member has a conflict of interest, is removed. This strong requirement is typically violated, because another PC member might, for example, nondeterministically post a message in a discussion about a paper where they both have no conflict. Weaker information flow policies, on the other hand, often turn out too weak. *Nondeducibility* [19], for example, only requires that an agent cannot deduce, i.e., conclusively determine, the classified information. The problem is that a piece of information is considered nondeducible already if, in the entire space of potential behaviors, there exists some other explanation. In reality, however, not all agents exhibit the full set of potentially possible behaviors, and an actual agent might be able to deduce far more than expected (cf. [15]).

*Temporal logics* for the specification of information flow [10] are an important step forward, because they make it possible to customize the secrecy properties. HyperLTL [7] is the linear-time representative of this class of logics. As an extension of linear-time temporal logic (LTL), HyperLTL can describe the precise circumstances under which a particular information flow policy must hold. While standard linear or branching-time logics, like LTL or CTL$^*$, can only reason about the observations at a single computation trace at a time, and can thus, by themselves, not specify information flow, HyperLTL formulas use trace quantifiers and trace variables to simultaneously refer to multiple traces. For example, HyperLTL can directly express information flow properties like "for any pair of traces $\pi, \pi'$, if the low-security observer sees the same inputs on $\pi$ and $\pi'$, then the low-security observer must also see the same outputs on $\pi$ and $\pi'$". The key limitation of HyperLTL for the specification of workflows is that it is a propositional logic. It is, hence, impossible to specify the information flow in workflows unless the number of agents is fixed *a-priori*. In this paper, we overcome this limitation.

We introduce a framework for the specification and verification of secrecy in workflows with arbitrarily many agents. Our framework consists of a workflow description language, a specification language, and a verification method. Our *workflow description language* gives a precise description of the behavior of workflow management systems with an arbitrary number of agents. Figure 1 shows a simple example workflow of a conference management system. The workflow manipulates several relations over the unbounded domain of agents, that each characterize a particular relationship between the agents: for example, a pair $(x, p)$ in *Conf* indicates that PC member $x$ has declared a conflict with paper $p$, a triple $(x, y, p)$ in *Comm* indicates that PC member $x$ has received from PC member $y$ a message about paper $p$. As a *specification language* for the information flow policies in such workflows, we introduce a first-order version of HyperLTL. We extend HyperLTL with first-order quantifiers, allowing the formulas to refer to an arbitrary number of agents. We show that the new logic can be used to specify precise assumptions on the behavior of the agents, such as *causality*: while a nondeterministic agent can take *any* action, the actions of a causal agent can only reveal *information the agent has actually observed.* Restricting the behaviors of the agents to the causal behavior allows us to quan-

```
(1) forall $x, p$. may $\mathit{true} \rightarrow \mathit{Conf} \mathrel{+}= (x, p)$
        % PC members declare conflicts of interest
(2) forall $x, p$. may $\neg \mathit{Conf}(x, p) \rightarrow A \mathrel{+}= (x, p)$
        % PC chair makes paper assignments taking into account the conflicts
(3) forall $x, p, r$. $A(x, p) \wedge \mathit{Oracle}(p, r) \rightarrow \mathit{Read} \mathrel{+}= (x, p, r)$
        % PC members without conflicts read reviews
(4) forall $y, x, p$. may $A(x, p) \wedge A(y, p) \rightarrow \mathit{Comm} \mathrel{+}= (x, y, p)$
        % PC discussion among members assigned to the same paper
```

**Fig. 1.** Example workflow from a conference management system.

tify universally over the actions of the agents, as in classic notions of secrecy like noninterference, and, at the same time, eliminate the false positives of these notions. Finally, we introduce a *verification method*, which translates the verification problem of workflows with arbitrarily many agents and specifications in first-order HyperLTL to the satisfiability problem of first-order logic. While first-order logic is in general undecidable, we identify conditions under which the satisfiability problem for the particular formulas in the verification of the workflows is guaranteed to be decidable.

## 2 Workflows with Arbitrarily Many Agents

**Symbolic Transition Systems.** As the formal setting for the specification and verification of our workflows, we chose symbolic transition systems, where the states are defined as the valuations of a set of first-order predicates $\mathcal{P}$. The initial states and the transitions between states are described symbolically using an assertion logic over $\mathcal{P}$. For the purpose of describing workflows, we use first-order predicate logic (PL) with equality as the assertion language.

A *symbolic transition system* $\mathcal{S} = (\mathcal{P}, \Theta, \Delta)$ consists of a set of predicates $\mathcal{P}$, an *initial condition* $\Theta$, and a *transition relation* $\Delta$. The initial condition $\Theta$ is given as a formula of the assertion language over the predicates $\mathcal{P}$. The transition relation $\Delta(P_1, \ldots, P_k; P'_1, \ldots, P'_k)$ is given as a formula over the predicates $\mathcal{P} = \{P_1, \ldots, P_k\}$, which indicate the interpretation of the predicates in the present state, and the set of primed predicates $\mathcal{P}' = \{P'_1, \ldots, P'_k\}$, which indicate the interpretation of the predicates in the next state.

Let $U$ be some arbitrary universe. In the case of the workflows, $U$ is the set of agents participating in the workflow. Let $\mathcal{P}^n$ denote the set of predicates with arity $n$. A *state* $s : \bigcup_{n \geq 0} \mathcal{P}^n \times U^n \rightarrow \mathbb{B}$ is then an evaluation of the predicates over $U$. A *trace* is an infinite sequence of states $s_0, s_1, \ldots$ such that (1) $s_0$ satisfies $\Theta$ (*initiation*), and (2) for each $i \geq 0$, the transition relation $\Delta$ is satisfied by the consecutive states $s_i$ and $s_{i+1}$, where the predicates in $\mathcal{P}$ are evaluated according to $s_i$ and the predicates in $\mathcal{P}'$ are evaluated according to $s_{i+1}$. We denote the set of all traces of a transition system $\mathcal{S}$ as $\mathit{Traces}(\mathcal{S})$.

**The Workflow Language.** We define a language to specify workflows. A workflow is structured into multiple *blocks*. Each block specifies the behaviour of a group of agents. A block is made of several statements which add (or remove) specific tuples from a given relation depending on a guard clause.

$$
\begin{array}{lcl@{\quad}l@{\quad}l}
p & ::= & \mathit{block}; p \mid \epsilon & // & \text{workflow program} \\
\mathit{block} & ::= & \textbf{forall } x_0, \ldots, x_k.\{\mathit{stmts}\} & & \\
 & & \mid \textbf{forall } x_0, \ldots, x_k.\ \textbf{may } \{\mathit{stmts}\} & // & \text{block} \\
\mathit{stmts} & ::= & \theta \to R \mathrel{+}= (t_1, \ldots, t_n); \mathit{stmts} & & \\
 & & \mid \theta \to R \mathrel{-}= (t_1, \ldots, t_n); \mathit{stmts} & & \\
 & & \mid \epsilon & // & \text{updates} \\
t & ::= & x_j \mid c & // & \text{terms}
\end{array}
$$

Here, terms $t_1, \ldots, t_n$ are either agent variables $x_0, \ldots, x_k$ or constant values $c$, $R$ denotes a predicate symbol, and $\theta$ is a *guard* clause that needs to be met before performing the update. If the guard is not met, no update occurs. Guards can be arbitrary formulas from first-order predicate logic (PL). The set of predicate symbols contains a special symbol *Oracle* denoting the environment input. In order to specify deterministic/nondeterministic behaviour, we use two different kinds of statements. In a normal block, all agents execute the block, i.e., the listed sequence of guarded updates. In a **may** block, only a subset of tuples of agents may decide to execute the block. Note that guarded remove to a predicate $R$ of the form $\theta \to R \mathrel{-}= (t_0, \ldots, t_n)$ can be simulated by a guarded addition to a fresh predicate $R'$. For that, we define: $R(t_0, \ldots, t_n) \wedge \neg\theta \to R' \mathrel{+}= (t_0, \ldots, t_n)$ and subsequently, replace uses of $R$ with uses of $R'$.

**Semantics.** In the following, we give a semantics for workflow $w = b_1 \ldots b_T$ as a transition system. The set of variables then consists of the universe $U$ of agents participating in the workflow, together with a finite set of relations or predicates over $U$. In order to control the transitions between system states, we require one predicate $\mathit{Choice}_i$ for the $i$-th **may** statement to control the subset of tuples of agents choosing to execute the statement. Furthermore, let $\mathit{Count}_0, \ldots, \mathit{Count}_T$ denote a sequence of boolean flags indicating the current program point. Iteration of the workflow from 0 to $T$ is expressed by the formula $\Phi_{\mathit{Count}}$ given by:
$\mathit{Count}_T \to (\mathit{Count}'_T \wedge \bigwedge_{l' \neq T} \neg \mathit{Count}'_{l'}) \wedge \bigwedge_{l=0}^{T-1} \mathit{Count}_l \to (\mathit{Count}'_{l+1} \wedge \bigwedge_{l' \neq l+1} \neg \mathit{Count}'_{l'})$
Initially, all predicates are *false*, except for the designated relation *Oracle* that provides input data to the workflow and the relations $\mathit{Choice}_i$ that provide the agent behaviour. Moreover, all flags $\mathit{Count}_l$, but $\mathit{Count}_0$ are *false*. An execution of the workflow program then is completely determined by the initial value of *Oracle* together with the choices of the agents as provided by the relations $\mathit{Choice}_i$. W.l.o.g., we assume that within a statement, every relation $R$ is updated at most once. For every $k$-ary relation $R$ and program point $l$, we construct a formula $\Phi_{R,l}(y_1, \ldots, y_k)$ using free variables $y_1, \ldots, y_k$, so that $R(y_1, \ldots, y_k)$ holds after execution of block $b_l$ iff $\Phi_{R,l}(y_1, \ldots, y_k)$ holds before the execution of $b_l$. The transition relation is defined by the conjunction of $\Phi_{\mathit{Count}}$ together with the conjunction over all formulas

$$
\begin{array}{l}
\bigwedge_{l=0}^{T-1} \mathit{Count}_l \to \forall y_1, \ldots, y_k.\, R'(y_1, \ldots, y_k) \leftrightarrow \Phi_{R,l}(y_1, \ldots, y_k) \wedge \\
\forall y_1, \ldots, y_k.\, \mathit{Count}_T \to R'(y_1, \ldots, y_k) \leftrightarrow R(y_1, \ldots, y_k)
\end{array}
$$

where $R'$ denotes the value of $R$ after the transition. Thus, we assume that after the last step, the workflow *stutters*, i.e., the last state is repeated indefinitely. For defining the formulas $\Phi_{R,l}(y_1,\ldots,y_k)$, consider a block $b_l$ of the form:

```
forall $x_0,\ldots,x_m$ {
    $\theta_1 \to R_1 \mathrel{+}= (t_{11},\ldots,t_{1k_1})$;
    ...
    $\theta_r \to R_r \mathrel{+}= (t_{r1},\ldots,t_{rk_r})$;
}
```

Then for $j = 1,\ldots,r$, $\Phi_{R_j,l}(y_1,\ldots,y_k)$ is $\Phi_{R_j,l}(y_1,\ldots,y_k) \vee \exists x_0,\ldots,x_m.\ \bar{\theta}_j \wedge (y_1 = t_{j1}) \wedge \ldots \wedge (y_{k_j} = t_{jk_j})$, where $\bar{\theta}_j$ is obtained from $\theta_j$ by replacing every literal $R_i(s_1,\ldots,s_{k_i})$ with the corresponding formula $\Phi_{R_i,l}(s_1,\ldots,s_{k_i})$. For all other predicates $R$, $\Phi_{R,l}(y_1,\ldots,y_k) \equiv \Phi_{R,l}(y_1,\ldots,y_k)$. If $b_l$ is the $n$-th **may** block and of the form:

```
forall $x_0,\ldots,x_m$ may {
    $\theta_1 \to R_1 \mathrel{+}= (t_{11},\ldots,t_{1k_1})$;
    ...
    $\theta_r \to R_r \mathrel{+}= (t_{r1},\ldots,t_{rk_r})$;
}
```

we proceed analogously, but add the choice relation $\mathit{Choice}_n(x_0,\ldots,x_m)$ as an additional condition to the $\theta_j$. Thus for $j = 1,\ldots,r$, $\Phi_{R_j,l}(y_1,\ldots,y_k)$ is given by:

$$\Phi_{R_j,l}(y_1,\ldots,y_k) \vee \exists x_1,\ldots,x_k.\ \bar{\theta}_j \wedge \mathit{Choice}_n(x_0,\ldots,x_m) \wedge (y_1 = t_{j1}) \wedge \ldots \wedge (y_{k_j} = t_{jk_j})$$

where $\bar{\theta}_j$ is obtained from $\theta_j$ by replacing every literal $R_i(s_1,\ldots,s_{k_i})$ with the corresponding formula $\Phi_{R_i,l}(s_1,\ldots,s_{k_i})$. For all other predicates $R$, $\Phi_{R,l}(y_1,\ldots,y_k) \equiv \Phi_{R,l}(y_1,\ldots,y_k)$.

We remark that, by successive substitution of the formulas $\Phi_{R,l}$, we obtain for every prefix of the workflow of length $l$ and for every predicate $R$, a formula $\bar{\Phi}_{R,l}$ which expresses the value of $R$ in terms of the predicates at program start and the predicates $\mathit{Choice}_i$ only.

*Example 1.* Consider a variation of the conference management workflow given in the introduction, where a set of all PC members that do not have a conflict with any paper is collected.

$(s_1)$ **forall** $x, p$ **may** $\mathit{true} \;\to\; \mathit{Conf} \mathrel{+}= (x,p)$
$(s_2)$ **forall** $x, p$ $\neg\mathit{Conf}(x,p) \;\to\; S \mathrel{+}= (x)$

Then for $(s_1)$, $\bar{\Phi}_{\mathit{Conf},1}(x,p) \equiv \bar{\Phi}_{\mathit{Conf},2}(x,p)$ is given by $\exists x_1, p_1.\ \mathit{Choice}_1(x_1,p_1) \wedge (x_1 = x) \wedge (p_1 = p)$, which is equivalent to $\mathit{Choice}_1(x,p)$. Accordingly for $(s_2)$, $\bar{\Phi}_{S,2}$ is given by: $\exists x_2, p_2.\ \neg\bar{\Phi}_{\mathit{Conf},1}(x_2,p_2) \wedge (x_2 = x)$ which can be simplified to $\exists p_2.\ \neg\bar{\Phi}_{\mathit{Conf},1}(x,p_2)$. Altogether, we obtain:

$$\begin{array}{ll} \bar{\Phi}_{\mathit{Conf},2}(x,p) & \equiv \mathit{Choice}_1(x,p) \\ \bar{\Phi}_{S,2}(x) & \equiv \exists p_2.\ \neg\mathit{Choice}_1(x,p_2) \end{array}$$

*Example 2.* Consider the workflow (WF1), shown in Fig. 1 in the introduction. Within this workflow, every statement updates exactly one predicate, and each predicate *Conf*, *A*, *Read* and *Comm* is updated only once. Accordingly, we can drop the extra index $t$ and write $\bar{\Phi}_{Conf}$, $\bar{\Phi}_A$, $\bar{\Phi}_{Read}$, $\bar{\Phi}_{Comm}$ for the corresponding predicates after their respective updates. We have:

$$\begin{aligned}
\bar{\Phi}_{Conf}(y_1, y_2) &\equiv \mathit{Choice}_1(y_1, y_2) \\
\bar{\Phi}_A(y_1, y_2) &\equiv \neg\mathit{Choice}_1(y_1, y_2) \wedge \mathit{Choice}_2(y_1, y_2) \\
\bar{\Phi}_{Read}(y_1, y_2, y_3) &\equiv \neg\mathit{Choice}_1(y_1, y_2) \wedge \mathit{Choice}_2(y_1, y_2) \wedge \mathit{Oracle}(y_2, y_3) \\
\bar{\Phi}_{Comm}(y_1, y_2, y_3) &\equiv \neg\mathit{Choice}_1(y_1, y_3) \wedge \mathit{Choice}_2(y_1, y_3) \wedge \\
&\quad\ \neg\mathit{Choice}_1(y_2, y_3) \wedge \mathit{Choice}_2(y_2, y_3) \wedge \mathit{Choice}_3(y_2, y_1, y_3)
\end{aligned}$$

□

## 3 Specifying Secrecy with First-Order HyperLTL

HyperLTL [7] is a recent extension of linear-time temporal logic (LTL) with *trace variables* and *trace quantifiers*. HyperLTL can express noninterference and other information flow policies by relating multiple traces, which are each identified by a separate trace variable. Since HyperLTL was introduced as a propositional logic, it cannot express properties about systems with an arbitrary number of agents. We now present *first-order* HyperLTL, which extends propositional HyperLTL with first-order quantifiers. In the following, we will refer to first-order HyperLTL simply as HyperLTL.

**HyperLTL syntax.** Let $\mathcal{P}$ be a set of predicates, $\mathcal{V}$ be a set of first-order variables, and $\Pi$ be a set of *trace variables*. We call the set $\mathcal{P}_\Pi = \{P_\pi \mid P \in \mathcal{P}, \pi \in \Pi\}$ the set of *indexed predicates*. Our logic builds on the assertion language used in the description of the symbolic transition systems. In the case of the workflows, this is first-order predicate logic (PL) with equality. The *atomic formulas* of HyperLTL are formulas of the assertion language over the indexed predicates $\mathcal{P}_\Pi$ and the variables $\mathcal{V}$. HyperLTL *formulas* are then generated by the following grammar (with initial symbol $\psi$):

$$\begin{aligned}
\psi &::= \exists\pi.\,\psi \mid \exists\pi.\,\varphi \mid \neg\psi \\
\varphi &::= \Psi \mid \neg\varphi \mid \varphi\wedge\varphi \mid \exists x.\,\varphi \mid \bigcirc\varphi \mid \varphi\ \mathcal{U}\ \varphi,
\end{aligned}$$

where $\Psi$ is an atomic formula, $\pi \in \Pi$ is a trace variable, and $x \in \mathcal{V}$ is a first-order variable. HyperLTL formulas thus start with a prefix of trace quantifiers consisting of at least one quantifier and then continue with a subformula that contains only first-order quantifiers, no trace quantifiers. Universal trace quantification is defined as $\forall\pi.\varphi \equiv \neg\exists\pi.\neg\varphi$. $\mathcal{U}$ and $\bigcirc$ are the usual Until and Next modalities from LTL. We also consider the usual derived Boolean operators and the derived temporal operators *Eventually* $\Diamond\varphi \equiv \mathit{true}\,\mathcal{U}\,\varphi$, *Globally* $\Box\varphi \equiv \neg\Diamond\neg\varphi$, and *Weak Until* $\varphi\ \mathcal{W}\ \psi \equiv \varphi\,\mathcal{U}\,\psi \ \vee\ \Box\varphi$.

**HyperLTL semantics.** The semantics of a HyperLTL formula $\psi$ is given with respect to a set of traces $\mathcal{T}$, an evaluation $\alpha : \mathcal{V} \to U$ of the first-order variables, and an evaluation $\beta : \Pi \to \mathcal{T}$ of the trace variables. Let $\sigma(n)$ denote the *n-th element* in a trace $\sigma$, and let $\sigma[n,\infty] = \sigma(n)\sigma(n+1)\ldots$ denote the *n-th suffix* of $\sigma$. We lift the suffix operation from traces to trace assignments and define $\beta[n,\infty](\sigma) := \beta(\sigma)[n,\infty]$. The *update* of an evaluation of the first-order or trace variables is defined as follows: $\gamma[x \mapsto a](x) = a$ and $\gamma[x \mapsto a](y) = \gamma(x)$ for $x \neq y$. The *satisfaction* of a HyperLTL formula $\psi$, denoted by $\alpha, \beta \models_{\mathcal{T}} \psi$, is then defined as follows:

$$\begin{array}{lll}
\alpha,\beta \models_{\mathcal{T}} \exists \pi.\ \psi & \text{iff} & \exists t \in \mathcal{T}.\ \alpha, \beta[\pi \mapsto t] \models_{\mathcal{T}} \psi, \\
\alpha,\beta \models_{\mathcal{T}} \neg\psi & \text{iff} & \alpha,\beta \not\models_{\mathcal{T}} \psi, \\
\alpha,\beta \models_{\mathcal{T}} \Psi & \text{iff} & \alpha,\delta \models \Psi, \\
\alpha,\beta \models_{\mathcal{T}} \varphi_1 \wedge \varphi_2 & \text{iff} & \alpha,\beta \models_{\mathcal{T}} \varphi_1 \text{ and } \alpha,\beta \models_{\mathcal{T}} \varphi_2, \\
\alpha,\beta \models_{\mathcal{T}} \exists x.\ \varphi & \text{iff} & \exists a \in U.\ \alpha[x \mapsto a], \beta \models_{\mathcal{T}} \varphi, \\
\alpha,\beta \models_{\mathcal{T}} \bigcirc\varphi & \text{iff} & \alpha,\beta[1,\infty] \models_{\mathcal{T}} \varphi, \\
\alpha,\beta \models_{\mathcal{T}} \varphi_1 \,\mathcal{U}\, \varphi_2 & \text{iff} & \exists i \geq 0 :\ \alpha,\beta[i,\infty] \models_{\mathcal{T}} \varphi_2 \text{ and} \\
 & & \forall 0 \leq j < i :\ \alpha,\beta[j,\infty] \models_{\mathcal{T}} \varphi_1,
\end{array}$$

where $\psi, \varphi_1$, and $\varphi_2$ are HyperLTL formulas, $\Psi$ is an atomic formula, and $\alpha, \delta \models \Psi$ denotes the satisfaction of the formula $\Psi$ of the assertion logic in the valuation $\alpha$ of the first-order variables and the interpretation $\delta$ of the indexed predicates. The interpretation $\delta(P_\pi)$ of an indexed predicate $P_\pi$ is defined as the interpretation $\delta(P_\pi) = \beta(\pi)(0)(P)$ of $P$ provided by the first state of the trace assigned to $\pi$. A formula without free first-order and trace variables is called *closed*. A closed formula $\psi$ is *satisfied* by a transition system $\mathcal{S}$, denoted by $\mathcal{S} \models \psi$, iff $\alpha, \beta \models_{\mathcal{T}} \psi$ for the empty assignments $\alpha$ and $\beta$ and the set $\mathcal{T} = \mathit{Traces}(\mathcal{S})$ of traces of the transition system. HyperLTL formulas in which all trace quantifiers are universal are called *universal formulas*. In the remainder of the paper, we will only consider universal formulas. This fragment contains many information flow properties of practical interest.

**Noninterference.** Secrecy properties like noninterference are based on a classification of the inputs and outputs of a system into either *low*, meaning not confidential, or *high*, meaning highly confidential. A system has the *noninterference* property [12] if in any pair of traces where the low inputs are the same, the low outputs are the same as well, regardless of the high inputs. When we are interested in the noninterference property of a single agent, it is possible to model the low and high inputs and the low and high outputs of the system (as seen by the agent) using separate predicates, for example, as $I_l, I_h, O_l, O_h$, respectively. Noninterference can then be expressed as the HyperLTL formula

$$\forall \pi. \forall \pi'.\ \Box(I_{l,\pi} \leftrightarrow I_{l,\pi'}) \ \to\ \Box(O_{l,\pi} \leftrightarrow O_{l,\pi'}),$$

which states that all traces $\pi$ and $\pi'$ that have the same low input $I_l$ at all times, must also have the same low output $O_l$ at all times.

In a workflow, the inputs or outputs of different agents may be collected in the same predicate. In the conference management example from the introduction, the low outputs observed by a PC member $x$ consist of the pairs $(x,p,r)$ for some paper $p$ in the *Read* relation and, additionally, of the tuples $(x,y,p)$ for some PC member $y$ and a paper $p$ in the *Comm* relation. The low input provided by agent $x$ is given by the tuples of the *Choice* predicates that begin with $x$. Additionally, the system has high input in the form of the *Oracle* predicate.

Generalizing from the example, we assume there is one or more predicates of the form $O_l(x,\vec{y})$, modeling *low output* observed by the agents from the system, and one or more predicates of the form $I_l(x,\vec{y})$ modeling *low inputs* provided by the agents to the system. An output is observable by agent $x$ whenever $x$ occurs in the first position of the tuple. Likewise, an input is controllable by agent $x$ whenever $x$ occurs in the first position of the tuple. The remaining components of the tuple are denoted by the vector $\vec{y} = y_1, y_2, \ldots$. Noninterference is then expressed as the HyperLTL formula

$$\forall \pi, \pi'. \forall x.\ \square(\forall \vec{y}\ .\ I_{l,\pi}(x,\vec{y}) \leftrightarrow I_{l,\pi'}(x,\vec{y})) \ \rightarrow\ \square(\forall \vec{y}.\ O_{l,\pi}(x,\vec{y}) \leftrightarrow O_{l,\pi'}(x,\vec{y}))$$

which states that, for all agents $x$, if the low input provided by agent $x$ on traces $\pi$ and $\pi'$ is the same, then the low output read by $x$ on $\pi$ and $\pi'$ must be the same as well.

**Declassification.** Declassification [18] becomes necessary when the functionality of the system makes it unavoidable that some information is leaked. In the conference management example, a PC member $x$ is supposed to read the reviews of the papers assigned to $x$. This is legitimate as long as $x$ has not declared a conflict of interest with those papers. We assume that, in addition to the input and output predicates, there is a *declassification condition* $D(x,\vec{y})$, which indicates that agent $x$ is allowed to learn about the high input $I_h(x,\vec{y})$. Noninterference with Declassification is then expressed as the HyperLTL formula

$$\begin{aligned}\forall \pi, \pi'. \forall x.\ &\square(\forall \vec{y}. I_{l,\pi}(x,\vec{y}) \leftrightarrow I_{l,\pi'}(x,\vec{y}) \ \wedge\ (D(x,\vec{y}) \rightarrow (I_{h,\pi}(x,\vec{y}) \leftrightarrow I_{h,\pi'}(x,\vec{y})))) \\ &\rightarrow \square(\forall \vec{y}.\ (O_{l,\pi}(x,\vec{y}) \leftrightarrow O_{l,\pi'}(x,\vec{y}))),\end{aligned}$$

which expresses that on all pairs of traces where the low inputs are the same and, additionally, the high inputs are the same whenever the declassification condition is true, the low outputs must be the same.

*Example 3.* In the conference management example, we specify the information flow policy that an agent should not receive information regarding papers where a conflict of interest has been declared as a noninterference property:

$$\begin{aligned}&\forall \pi, \pi'. \forall x.\ \square(\forall \vec{y}. \textstyle\bigwedge_{i=1}^{3}(\mathit{Choice}_{i,\pi}(x,\vec{y}) \leftrightarrow \mathit{Choice}_{i,\pi'}(x,\vec{y})) \wedge \\ &(\forall p, r.\ (\neg \mathit{Conf}_{\pi}(x,p) \wedge \neg \mathit{Conf}_{\pi'}(x,p)) \rightarrow (\mathit{Oracle}_{\pi}(p,r) \leftrightarrow \mathit{Oracle}_{\pi'}(p,r)))) \ \rightarrow \\ &\square(\forall p, r.\ (\mathit{Read}_{\pi}(x,p,r) \leftrightarrow \mathit{Read}_{\pi'}(x,p,r)) \wedge (\forall y, p.\ \mathit{Comm}_{\pi}(x,y,p) \leftrightarrow \mathit{Comm}_{\pi'}(x,y,p)))\end{aligned}$$

□

**Causality assumptions on agents.** In the workflow from Fig. 1, it is easy to see that no PC member can directly read the reviews of papers where a conflict of interest has been declared: the PC member can only read a review if the PC member was assigned to the paper, which, in turn, can only happen if no conflict of interest was declared. It is much more difficult to rule out an indirect flow of information via a message sent by another PC member. So far, neither the description of the workflow, nor the HyperLTL specification would prevent other PC members to add such messages to *Comm.* To rule out messages that would leak information about papers where a PC member has a conflict, we must make assumptions about the possible behaviors of the *other* agents.

*Stubborn agents.* A radical restriction on the behavior of the other agents is to require that they always, stubbornly, produce the same input, independently of their own observations. We assume that the input is represented by one or more predicates of the form $I(x, \vec{y})$, where an input is controllable by agent $x$ whenever $x$ occurs in the first position of the tuple. The requirement for traces $\pi, \pi'$ that all agents are *stubborn* can be specified by the HyperLTL formula:

$$\forall x.\ \square(\forall \vec{y}.\ I_\pi(x, \vec{y}) \leftrightarrow I_{\pi'}(x, \vec{y})).$$

*Causal agents.* A more natural restriction on the behavior of the other agents is to require that they act causally, i.e., they only provide different inputs if they, themselves, have previously observed different outputs. The *causality* of agents w.r.t. traces $\pi, \pi'$ can be described by the HyperLTL formula:

$$\forall x.\ (\forall \vec{y}.\ I_\pi(x, \vec{y}) \leftrightarrow I_{\pi'}(x, \vec{y}))\ \ \mathcal{W}\ \ (\exists \vec{y}.\ O_\pi(x, \vec{y}) \not\leftrightarrow O_{\pi'}(x, \vec{y}))$$

which states that, for all agents $x$ the inputs provided on two traces are the same *until* a difference in the outputs observed by $x$ occurs.

*Example 4.* In the conference management example, stubbornness for traces $\pi, \pi'$ can be specified as the HyperLTL formula $\forall x.\ \square(\forall y.\ \mathit{Choice}_{1,\pi}(x, y) \leftrightarrow \mathit{Choice}_{1,\pi'}(x, y) \wedge \ldots)$. The requirement of causality for $\pi, \pi'$ is specified as the HyperLTL formula

$$\begin{aligned}&\forall x.\ (\forall y.\ \mathit{Choice}_{1,\pi}(x, y) \leftrightarrow \mathit{Choice}_{1,\pi'}(x, y) \wedge \ldots)\ \ \mathcal{W}\\ &((\exists p, r.\ \mathit{Read}_\pi(x, p, r) \not\leftrightarrow \mathit{Read}_{\pi'}(x, p, r)) \vee (\exists y, p.\ \mathit{Comm}_\pi(x, y, p) \not\leftrightarrow \mathit{Comm}_{\pi'}(x, y, p))).\end{aligned}$$

□

Combining the agent assumptions with the specification of noninterference (and possibly declassification), we obtain a formula of the form

$$\forall \pi_1, \ldots, \pi_n.\ \varphi_{\mathsf{causal}} \rightarrow \varphi,$$

where $\varphi_{\mathsf{causal}}$ describes the agent assumption on all pairs of paths in $\pi_1, \ldots, \pi_n$.

## 4 Verifying Secrecy

We now present a bounded model checking method for symbolic transition systems and HyperLTL specifications. The approach reduces the violation of a HyperLTL formula on the prefix of a trace of a given symbolic transition system to the satisfiability of a formula of the assertion language. For workflows, it suffices to consider prefixes of bounded length, because the workflow terminates (and then stutters forever) after a fixed number of steps. Since the assertion language in the description of the workflows is first-order predicate logic, satisfiability of the resulting formula is not necessarily decidable. We return to this issue in Section 5, where we identify conditions under which decidability is guaranteed.

**Bounded Satisfaction.** Bounded model checking is based on a restricted notion of HyperLTL satisfaction where only trace prefixes of length $n$, for some fixed bound $n$, are considered. Let $\mathcal{T}$ be a set of traces, $\alpha : \mathcal{V} \to U$ an evaluation of the first-order variables, and $\beta : \Pi \to \mathcal{T}$ an evaluation of the trace variables. The *$n$-bounded satisfaction* of a HyperLTL formula $\psi$, denoted by $\alpha, \beta \models^n_{\mathcal{T}} \psi$, is then defined as follows:

$$
\begin{array}{lll}
\alpha, \beta \models^n_{\mathcal{T}} \exists \pi.\ \psi & \text{iff} & \exists t \in \mathcal{T}.\ \alpha, \beta[\pi \mapsto t] \models^n_{\mathcal{T}} \psi, \\
\alpha, \beta \models^n_{\mathcal{T}} \neg\psi & \text{iff} & \alpha, \beta \not\models^n_{\mathcal{T}} \psi, \\
\alpha, \beta \models^n_{\mathcal{T}} \Psi & \text{iff} & \alpha, \delta \models \Psi, \\
\alpha, \beta \models^n_{\mathcal{T}} \varphi_1 \wedge \varphi_2 & \text{iff} & \alpha, \beta \models^n_{\mathcal{T}} \varphi_1 \text{ and } \alpha, \beta \models^n_{\mathcal{T}} \varphi_2, \\
\alpha, \beta \models^n_{\mathcal{T}} \exists x.\ \varphi & \text{iff} & \exists a \in U.\ \alpha[x \mapsto a], \beta \models^n_{\mathcal{T}} \varphi, \\
\alpha, \beta \models^n_{\mathcal{T}} \bigcirc\varphi & \text{iff} & \alpha, \beta[1,\infty] \models^{n-1}_{\mathcal{T}} \varphi, \text{ for } n > 0, \\
\alpha, \beta \models^0_{\mathcal{T}} \bigcirc\varphi & \text{iff} & \alpha, \beta \models_{\mathcal{T}} \varphi, \\
\alpha, \beta \models^n_{\mathcal{T}} \varphi_1\, \mathcal{U}\, \varphi_2 & \text{iff} & \exists i \geq 0 :\ \alpha, \beta[i,\infty] \models^{n-i}_{\mathcal{T}} \varphi_2 \text{ and} \\
 & & \forall 0 \leq j < i :\ \alpha, \beta[j,\infty] \models^{n-j}_{\mathcal{T}} \varphi_1, \text{ for } n > 0, \\
\alpha, \beta \models^0_{\mathcal{T}} \varphi_1\, \mathcal{U}\, \varphi_2 & \text{iff} & \alpha, \beta[i,\infty] \models^0_{\mathcal{T}} \varphi_2,
\end{array}
$$

where $\psi, \varphi, \varphi_1$, and $\varphi_2$ are HyperLTL formulas, $\Psi$ is an atomic formula, and $\delta(P_\pi) = \beta(\pi)(0)(P)$. A closed formula $\psi$ is *$n$-bounded satisfied* by a transition system $\mathcal{S}$, denoted by $\mathcal{S} \models^n \psi$, iff $\alpha, \beta \models^n_{\mathcal{T}} \psi$ for the empty assignments $\alpha$ and $\beta$ and the set $\mathcal{T} = \mathit{Traces}(\mathcal{S})$ of traces of the transition system.

For workflows, satisfaction and bounded satisfaction coincide.

**Theorem 1.** *Let $\mathcal{S}$ be the transition system representing a workflow with $n$ blocks. For all HyperLTL formulas $\psi$, it holds that $\mathcal{S} \models \psi$ iff $\mathcal{S} \models^n \psi$.*

**Bounded Model Checking.** We now translate a transition system $\mathcal{S}$ and a given universal HyperLTL formula for a given bound $n$ into a formula $\Psi_{\mathcal{S},\neg\psi}$ of the assertion language such that $\Psi_{\mathcal{S},\neg\psi}$ is satisfiable iff $\mathcal{S} \not\models^n \psi$. Since $\psi$ is universal, its negation is of the form $\exists \pi_1, \ldots, \pi_k.\ \varphi$, where $\varphi$ does not contain any more trace quantifiers. Let the set of predicates $\mathcal{P}$ be given as $\mathcal{P} = \{P_1, \ldots, P_m\}$. In $\Psi^n_{\mathcal{S},\neg\psi}$, we use for every predicate $P_i$ several copies $P_{i,\pi,l}$, one per trace variable $\pi \in \{\pi_1, \ldots, \pi_k\}$ and position $l$, $0 \leq l \leq n$. The formula $\Psi^n_{\mathcal{S},\neg\psi} = [\![\mathcal{S}]\!]^n \wedge [\![\varphi]\!]^n_0$ is a conjunction of two formulas of the assertion language, the unfolding $[\![\mathcal{S}]\!]^n$ of the transition system $\mathcal{S}$ and the unfolding $[\![\varphi]\!]^n_0$ of the HyperLTL formula $\varphi$.

For a symbolic transition system $\mathcal{S}$ and a bound $n \geq 0$, the *unfolding* $[\![\mathcal{S}]\!]^n$ is defined as follows:

$$[\![\mathcal{S}]\!]^n = \bigwedge_{\pi \in \{\pi_1, \ldots, \pi_k\}} \Theta(P_{1,\pi,0}, \ldots P_{m,\pi,0}) \wedge \bigwedge_{l=0}^{n-1} \Delta(P_{1,\pi,l}, \ldots, P_{k,\pi,l}; P_{1,\pi,l+1}, \ldots, P_{m,\pi,l+1})$$

For a HyperLTL formula $\varphi$ without trace quantifiers and a bound $n \geq 0$, the *unfolding* $[\![\varphi]\!]_l^n$ is defined as follows:

$$\begin{array}{lcl}
[\![\neg\varphi]\!]_l^n & = & \neg[\![\varphi]\!]_l^n, \\
[\![\Psi]\!]_l^n & = & \Psi_l, \\
[\![\varphi_1 \wedge \varphi_2]\!]_l^n & = & [\![\varphi_1]\!]_l^n \wedge [\![\varphi_2]\!]_l^n, \\
[\![\exists x.\ \varphi]\!]_l^n & = & \exists x.\ [\![\varphi]\!]_l^n, \\
[\![\bigcirc\varphi]\!]_l^n & = & [\![\varphi]\!]_{l+1}^{n-1} \text{ for } n > 0, \\
[\![\bigcirc\varphi]\!]_l^0 & = & [\![\varphi]\!]_l^0, \\
[\![\varphi_1 \,\mathcal{U}\, \varphi_2]\!]_l^n & = & [\![\varphi_2]\!]_l^n \vee ([\![\varphi_1]\!]_l^n \ \wedge [\![\varphi_1 \,\mathcal{U}\, \varphi_2]\!]_{l+1}^{n-1}) \text{ for } n > 0, \\
[\![\varphi_1 \,\mathcal{U}\, \varphi_2]\!]_l^0 & = & [\![\varphi_2]\!]_l^0
\end{array}$$

where $\varphi, \varphi_1$, and $\varphi_2$ are HyperLTL formulas, $\Psi$ is a formula of the assertion language over indexed predicates $P_{i,\pi}$ and $\Psi_l$ is the same formula with all occurrences of an indexed predicate $P_{i,\pi}$ replaced by the predicate $P_{i,\pi,l}$.

**Theorem 2.** *For a symbolic transition system $S$, a universal HyperLTL formulas $\psi$, and a bound $n \geq 0$, it holds that $S \models^n \psi$ iff $\Psi^n_{\mathcal{S},\neg\psi}$ is unsatisfiable.*

Combining Theorems 1 and 2, we obtain the corollary that bounded model checking is a complete verification technique for workflows.

**Corollary 1.** *Let $\mathcal{S}$ be the transition system representing a workflow with $T$ blocks. For all HyperLTL formulas $\psi$, it holds that $\mathcal{S} \models \psi$ iff $\Psi^T_{\mathcal{S},\neg\psi}$ is unsatisfiable.* □

## 5 Decidability

We now identify cases where the satisfiability of the predicate logic formulas constructed by the verification method of the previous section are decidable. For background on PL and decidable subclasses, we refer to the textbook [6].

**Theorem 3.** *Consider a workflow consisting of $T$ blocks where all agents are* stubborn*, and every predicate $R$ encountered by the workflow after $l$ blocks, is characterized by a quantifier-free formula $\bar{\Phi}_{R,l}$. Assume that $\forall\pi_1, \ldots, \pi_r.\ \varphi_{\mathsf{stubborn}} \rightarrow \varphi$ denotes a HyperLTL formula where $\Psi' \equiv [\![\neg\varphi]\!]_0^T$ is a Bernays-Schönfinkel formula, i.e., the prenex form of $\Psi'$ has a quantifier sequence of the form $\exists^*\forall^*$. Then it is decidable whether $\forall\pi_1, \ldots, \pi_r.\ \varphi_{\mathsf{stubborn}} \rightarrow \varphi$ holds.*

*Proof.* For every predicate $R$, let $\bar{\Phi}_{R,\pi_j,l}$ denote the formula which characterizes $R_{\pi_j,l}$, i.e., the value of $R$ after $l$ blocks along the execution of $\pi_j$. The formula $\bar{\Phi}_{R,\pi_j,l}$ is obtained from $\bar{\Phi}_{R,l}$ by replacing the occurrences of $Choice_i$, $Oracle$ with $Choice_{i,\pi_j}$ and $Oracle_{\pi_j}$, respectively. Let $\bar{\Psi}'$ denote the formula obtained from $\Psi'$ by first replacing every occurrence of a literal $R_{\pi_j,l}(s_1, \ldots, s_k)$ with $\bar{\Phi}_{R,\pi_j,l}(s_1, \ldots, s_k)$. As all agents are stubborn, the predicates $Choice_{i,\pi_j}$ are equivalent for $j = 1, \ldots, r$. Accordingly, we may replace all $Choice_{i,\pi_j}(s_1, \ldots, s_k)$

with $\mathit{Choice}_{i,\pi_1}(s_1,\ldots,s_k)$. The resulting formula is still a Bernays-Schönfinkel formula. It is unsatisfiable iff $\forall\pi_1,\ldots,\pi_r.\ \varphi_{\mathsf{stubborn}} \to \varphi$ is universally true. Satisfiability of $\bar{\Psi}'$, however, is decidable — which thus implies the the theorem. $\square$

Theorem 3 can be extended also to more general classes of workflows, given that the predicates $R_{\pi_j,l}$ occur only positively or only negatively in $\Psi'$. Non-interference, however, amounts to stating that (under certain conditions) no distinction is observable between some $R_{\pi_j,l}$ and $R_{\pi_{j'},l}$. Logically, indistinguishability is expressed by equivalence, which thus results in both positive and negative occurrences of the predicates in question.

**Theorem 4.** *Consider a workflow consisting of $T$ blocks where all agents are* causal*, and every predicate $R$ encountered by the workflow after $l$ blocks, is characterized by a quantifierfree formula $\bar{\Phi}_{R,l}$. Assume that $\forall\pi_1,\ldots,\pi_r.\ \varphi_{\mathsf{causal}} \to \varphi$ is a temporal formula where the prenex form of $\Psi' \equiv [\![\neg\varphi]\!]_0^T$ is purely existential. Then it is decidable whether $\forall\pi_1,\ldots,\pi_r.\ \varphi_{\mathsf{causal}} \to \varphi$ holds.*

*Proof.* The argument for causal agents is somewhat more complicated and accordingly leads to decidability only for a smaller fragment of HyperLTL formulas. Removal of the temporal operators and skolemization of the formula $\varphi_{\mathsf{causal}}$ describing causality yields a conjunction of clauses of form $(*)$: $S(x, f_1(x),\ldots,f_r(x)) \vee \mathit{Choice}_{i,\pi_{j_1}}(x,\mathbf{z}) \vee \neg\mathit{Choice}_{i,\pi_j}(x,\mathbf{z})$ or $S(x, f_1(x),\ldots,f_r(x)) \vee \neg\mathit{Choice}_{i,\pi_{j_2}}(x,\mathbf{z}) \vee \mathit{Choice}_{i,\pi_j}(x,\mathbf{z})$, for $j_1, j_2 < j$, where the disjunction $S$ refers to predicates which depend on $\mathit{Choice}_{i',\_}$ predicates for $i' < i$ only. In order to perform ordered resolution, we put an ordering upon predicates so that $\mathit{Choice}_{i,\pi_j}$ receives a higher priority than $\mathit{Choice}_{i',\pi_{j'}}$ if $i' < i$ or, if $i = i'$, $j' < j$. Moreover all predicates in $S$ have lower priorities than the $\mathit{Choice}$ predicates. Accordingly, the highest priority literal in each clause of $\varphi_{\mathsf{causal}}$ contains all free variables of the clause.

Let us first consider the case $r = 2$. Then resolution of two clauses with a positive and negative occurrence of the same highest-priority literal will result in a tautology and therefore is useless. As in the proof of Theorem 3, let $\bar{\Psi}'$ denote the formula obtained from $\Psi'$ by replacing each occurrence of a predicate $R_{\pi_j,l}(s_1,\ldots,s_k)$ with the formulas $\bar{\Phi}_{R,\pi_j,l}(s_1,\ldots,s_k)$ $(j = 1, 2)$. According to our assumption on $\Psi'$, the clauses obtained from $\bar{\Psi}'$ are all ground. Resolution of such a clause with a clause of $\varphi_{\mathsf{causal}}$ for some $\mathit{Choice}_{i,\pi_2}$ will again return a ground formula. By substituting the semantic formulas $\bar{\Phi}_{R,\pi_j,l}$ we obtain a set of new ground clauses, this time, however, with occurrences of predicates $\mathit{Choice}_{i',\pi_{j'}}, i' < i$, only. As a consequence, for every $i$, there is a bounded number of new clauses derivable by means of clauses from $\varphi_{\mathsf{causal}}$ with highest priority predicate $\mathit{Choice}_{i,\pi_2}$. Altogether, we therefore obtain only a bounded number of ground clauses which are derivable by means of ordered resolution. Hence, it is decidable whether a contradiction is derivable or not. This concludes the proof.

The argument for $r > 2$ is similar, only that resolution of any two such clauses originating from $\varphi_{\mathsf{causal}}$ with $j_1 \neq j_2$ upon the literal $\mathit{Choice}_{i,\pi_j}(x,\mathbf{z})$) will again result in a clause of the given form. In particular, no further literals are introduced. Therefore, saturation of $\varphi_{\mathsf{causal}}$ by ordered resolution will eventually terminate. Then the argument for termination proceeds analogously to the case $r = 2$ where $\varphi_{\mathsf{causal}}$ is replaced with the saturation of $\varphi_{\mathsf{causal}}$. $\square$

Theorem 4 can be extended to formulas $\varphi$ where $\bar{\Psi}'$ obtained from $[\![\neg\varphi]\!]_0^T$ is a Bernays-Schönfinkel formula at least in restricted cases.

Consider the clauses of the form $(*)$ as obtained from $\varphi_{\mathsf{causal}}$ after skolemization. In case that the disjunction $S$ is empty, we call the corresponding clause *simple*, otherwise *complex*. Now assume that complex clauses from the saturation of $\varphi_{\mathsf{causal}}$ are always resolved with clauses (originating from the skolemization of $\bar{\Psi}'$) upon a *ground* literal. Then the same argument as in the proof of Theorem 4 applies to show that saturation by resolution will eventually terminate.

## 6 Completing the Conference Management Example

We now complete the verification of our running example, that no PC member learns about the reviews of a paper for which he has declared a conflict. As already discussed in Section 3, it is easy to see that no PC member can directly read the reviews of papers where a conflict of interest has been declared. To prove the noninterference property in Example 3, it remains to show that the communication received from the other agents is the same on two traces $\pi$ and $\pi'$ whenever the Oracle for the papers with a conflict are the same on $\pi$ and $\pi'$.

For both stubborn and causal agents, the predicates $\mathit{Conf}_\pi(x,p)$ and $A_\pi(y,p)$ coincide with their counterparts in $\pi'$. Furthermore, for stubborn agents, the *Choice* predicates do not depend on the execution paths. As the predicates $\mathit{Comm}_\pi$ and $\mathit{Comm}_{\pi'}$ only depend on *Choice* predicates, the equivalence in the conclusion is trivially true. Hence, the property holds under the assumption that the agents are stubborn.

The situation is different for causal agents. The causality assumption $\varphi_{\mathsf{causal}}$ (given in Section 3) states that the other PC members only send different communications if there was a different observation on the two traces. Since causality already implies that $\mathit{Choice}_1$ and $\mathit{Choice}_2$ are equal on all paths, this can be omitted from the antecendent of the requirement. The negation of the remaining property is then given by the following formula:

$$\begin{aligned}&\exists\pi,\pi'.\,\varphi_{\mathsf{causal}} \wedge \forall x.\,\square(\forall y,p.\ \mathit{Choice}_3(x,y,p) \leftrightarrow \mathit{Choice}_3(x,y,p)\wedge\\&(\forall p,r.\ (\neg\mathit{Conf}_\pi(x,p) \wedge \neg\mathit{Conf}_{\pi'}(x,p)) \to \mathit{Oracle}_\pi(p,r) \leftrightarrow \mathit{Oracle}_{\pi'}(p,r))) \wedge\\&\Diamond(\exists y,p.\ \mathit{Comm}_\pi(x,y,p) \not\leftrightarrow \mathit{Comm}_{\pi'}(x,y,p))\end{aligned}$$

Due to the causality assumption, when we unroll $\mathcal{W}$ and replace *Read* with its semantics formula $\bar{\Phi}_{\mathit{Read}}$, we obtain that $\mathit{Choice}_1$ and $\mathit{Choice}_2$ are always equal and $\mathit{Choice}_3$ could differ on $\pi$ and $\pi'$ if there is a difference in the oracle.

$$\begin{aligned}\bar{\varphi}_{\mathsf{causal}} = {} & \forall x,p.\ \mathit{Choice}_{1,\pi}(x,p) \leftrightarrow \mathit{Choice}_{1,\pi'}(x,p)\ \wedge\ \forall x,p.\ \mathit{Choice}_{2,\pi}(x,p) \leftrightarrow \mathit{Choice}_{2,\pi'}(x,p)\ \wedge\\ & \forall x,y,p,r.\ (\mathit{Oracle}_\pi(p,r) \leftrightarrow \mathit{Oracle}_{\pi'}(p,r))\ \to\ (\mathit{Choice}_{3,\pi}(x,y,p) \leftrightarrow \mathit{Choice}_{3,\pi'}(x,y,p))\end{aligned}$$

Since in our example, the relation *Comm* is only assigned once, the $\Diamond$ operator is unrolled to a large disjunction that is false everywhere before the last step, since

*Comm* is empty on both paths. By unrolling $\neg\Psi$ and subsequently simplifying the formula with the causal equalities for $Choice_1$ and $Choice_2$, we obtain:

$$\begin{aligned}\neg\bar{\Psi} = \exists \pi, \pi'.\bar{\varphi}_{\mathsf{causal}} \wedge \exists x.(\forall p, r.\ &Choice_{1,\pi}(x,p)\ \vee Oracle_{\pi}(p,r) \leftrightarrow Oracle_{\pi'}(p,r)) \wedge \\ &\exists y, p'.\ Choice_{3,\pi}(y,x,p') \not\leftrightarrow Choice_{3,\pi'}(y,x,p')\end{aligned}$$

Note that all literals $Choice_{3,\pi}(y,x,p')$, $Choice_{3,\pi'}(y,x,p')$ contain existentially quantified variables only. Therefore, the assumptions of (the extension of) Theorem 4 are met. For the given formula, no contradiction can be derived. Instead, a model can be constructed as follows:

$$\begin{array}{llll} U & = \{x, y, p_1, p_2, r\}, & & \\ Oracle_{\pi} & = \{(p_1, r)\} & Oracle_{\pi'} & = \emptyset \\ Choice_{1,\pi} & = \{(x, p_1)\} & Choice_{1,\pi'} & = \{(x, p_1)\}, \\ Choice_{2,\pi} & = \{(x, p_2), (y, p_1), (y, p_2)\} & Choice_{2,\pi'} & = \{(x, p_2), (y, p_1), (y, p_2)\}, \\ Choice_{3,\pi} & = \{(y, x, p_1)\} & Choice_{3,\pi'} & = \emptyset \end{array}$$

Suppose the PC member $x$ who has a conflict with paper $p$ is assigned to a paper $q$ where he does not have a conflict, and another PC member $y$, who does not have a conflict with either paper, is assigned to both $p$ and $q$. Then $y$ can communicate with $x$ and therefore leak the review on paper $p$ to $x$.

To repair the problem, we let the PC chair remove the assignment of PC member $y$ to paper $q$ in such situations. Let (WF2) be (WF1) with the new line (2a) added in-between lines (2) and (3):

(2a) **forall** $x, y, p, q.\ Conf(x,p) \wedge \neg Conf(y,p) \wedge A(x,q) \wedge A(y,q) \rightarrow A \mathrel{-}= (y,q)$
  *% PC chair removes assignments that might cause leaks*

For the resulting workflow (WF2), we obtain a new formula $\bar{\Phi}_{A'}$, which in turn affects the formulas $\bar{\Phi}_{Read}$ and $\bar{\Phi}_{Comm}$ for *Read* and *Comm*:

$$\begin{aligned}\bar{\Phi}_{A'}(y,q) = \neg Choice_1(y,q) \wedge Choice_2(y,q) \wedge \\ \forall x, p.\ (Choice_1(x,q) \vee \neg Choice_2(x,q) \vee \neg Choice_1(x,p) \vee Choice_1(y,p))\end{aligned}$$

The resulting formula after substitution of the semantics formulas and simplification is similar to $\neg\bar{\Psi}$, but adds two conjunctions with the $\forall$-clause of $\bar{\Phi}_{A'}$ instantiated for $(x,p')$ and $(y,p')$ on both sides of the inequivalence. The resulting formula is a Bernays-Schönfinkel formula where again the decision procedure of Theorem 4 can be applied. That procedure now derives a contradiction. Intuitively, the reason is that on both paths, $x$ has declared a conflict with $p_1$. Since $y$ is assigned to $p_1$, $x$ and $y$ cannot be assigned jointly to the same paper. Thus, both sides of the inequivalence collapse to *false* — implying that for (WF2) requirement (2) is satisfied and thus (WF2) is indeed noninterferent. $\square$

## 7 Related Work

There is a vast body of work on information flow policies and associated verification techniques. We mention Goguen and Meseguer's seminal work on *noninter-*

*ference* [12], Zdancewic and Myer's *observational determinism* [20], Sutherland's *nondeducability* [19], and Halpern and O'Neill's *secrecy maintenance* [13] as representative examples. See Kanav *et al.* [15] for a recent overview with a detailed discussion of the most relevant notions for the verification of workflows. Our approach is based on the temporal logic HyperLTL [7]. HyperLTL has been applied in the verification of hardware systems, such as an Ethernet controller with 20000 latches [11]. Other logical approaches to information flow control include SecLTL [8], the polyadic modal $\mu$-calculus [2] and the epistemic temporal logics [9]. While standard linear-time temporal logic has been extended with first-order quantifiers [16], our first-order extension of HyperLTL is the first temporal logic for the specification of information flow in systems with arbitrarily many agents. In terms of practical verification efforts, there has been a lot of recent interest in proving secrecy in web-based workflow management systems. For example, for the ConfiChair conference management system it was proven that the system provider cannot learn the contents of papers [3]. For CoCon, another conference management system, it was proven that various groups of users, such as authors, reviewers, and PC members cannot deduce certain content, such as reviews, unless certain declassification triggers, such as being a PC member without a conflict of interest, are met [15]. For the verification of an eHealth system, Bhardwaj and Prasad [5] assume that all agents are known at analysis time. Based on this information, the authors construct a dedicated security lattice and then apply techniques from universal information flow [14, 1]. Our verification method is based on a reduction to the satisfiability problem of first-order predicate logic. First-order logic has many applications in verification. Most related, perhaps, is recent work on the verification of software defined networks [4, 17]. There, a network controller is translated into a first order formula and either a theorem prover or an SMT-solver is used to determine properties of the topology so that the controller satisfies a given invariant.

## 8 Conclusion

We have presented a formalization of fine-grained security properties for workflow systems with an unbounded number of agents. HyperLTL is the first approach to specify hyperproperties for systems without a fixed set of agents. For the verification of HyperLTL formulas, we have provided a bounded model checking algorithm that translates the problem of verifying such a property for a given workflow to the satisfiability of first-order predicate logic. We have also provided a non-trivial fragment of properties and workflows so that the corresponding decision problem is decidable. As an example we considered noninterference for a simple workflow of a conference management system. Unexpectedly, our method exhibited a subtle form of indirect information flow. We also indicated how that deficiency can be cured. All corresponding proving took place within our benevolent fragments. Various problems remain for future work. Further decidable fragments are of major concern. Also, our work should be extended to more complex and thus more expressive forms of workflows.

*Acknowledgment.* This work was partially supported by the German Research Foundation (DFG) under the project “SpAGAT” (grant no. FI 936/2-1) in the priority program “Reliably Secure Software Systems – RS3” and the doctorate program “Program and Model Analysis - PUMA” (no. 1480).

## References

1. Amtoft, T., Banerjee, A.: Information flow analysis in logical form. In: Giacobazzi, R. (ed.) Proc. SAS 2004. pp. 100–115. Springer (2004)
2. Andersen, H.R.: A polyadic modal $\mu$-calculus. Tech. rep., Danmarks Tekniske Universitet (1994)
3. Arapinis, M., Bursuc, S., Ryan, M.: Privacy supporting cloud computing: Confichair, a case study. In: Proc. POST 2012. pp. 89–108. Springer Verlag (2012)
4. Ball, T., Bjørner, N., Gember, A., Itzhaky, S., Karbyshev, A., Sagiv, M., Schapira, M., Valadarsky, A.: Vericon: towards verifying controller programs in software-defined networks. In: ACM SIGPLAN Notices. vol. 49, pp. 282–293. ACM (2014)
5. Bhardwaj, C., Prasad, S.: Parametric information flow control in ehealth. In: Proceedings HealthCom 2015. pp. 102–107 (Oct 2015)
6. Börger, E., Grädel, E., Gurevich, Y.: The Classical Decision Problem. Perspectives in Mathematical Logic, Springer (1997)
7. Clarkson, M., Finkbeiner, B., Koleini, M., Micinski, K.K., Rabe, M.N., Sánchez, C.: Temporal logics for hyperproperties. In: Proc. of POST (2014)
8. Dimitrova, R., Finkbeiner, B., Kovács, M., Rabe, M.N., Seidl, H.: Model checking information flow in reactive systems. In: Proc. VMCAI’12. pp. 169–185 (2012)
9. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. MIT Press (1995)
10. Finkbeiner, B., Rabe, M.N.: The linear-hyper-branching spectrum of temporal logics. it - Information Technology 56(6), 273–279 (2014)
11. Finkbeiner, B., Rabe, M.N., Sánchez, C.: Algorithms for model checking HyperLTL and HyperCTL$^*$. In: Computer Aided Verification. pp. 30–48. Springer (2015)
12. Goguen, J.A., Meseguer, J.: Security policies and security models. In: IEEE Symp. on Security and Privacy. pp. 11–20 (1982)
13. Halpern, J.Y., O’Neill, K.R.: Secrecy in multiagent systems. ACM Trans. Inf. Syst. Secur. 12(1), 5:1–5:47 (Oct 2008)
14. Hunt, S., Sands, D.: On flow-sensitive security types. In: Morrisett, J.G., Jones, S.L.P. (eds.) Proc. POPL 2006. pp. 79–90 (2006)
15. Kanav, S., Lammich, P., Popescu, A.: A conference management system with verified document confidentiality. In: CAV 2014. pp. 167–183. Springer Verlag (2014)
16. Manna, Z., Pnueli, A.: Verification of concurrent programs: The temporal framework. In: Boyer, R.S., Moore, J.S. (eds.) The Correctness Problem in Computer Science, pp. 215–273. Academic Press, London (1981)
17. Padon, O., Immerman, N., Karbyshev, A., Lahav, O., Sagiv, M., Shoham, S.: Decentralizing SDN policies. In: ACM SIGPLAN Notices. vol. 50, pp. 663–676. ACM (2015)
18. Sabelfeld, A., Sands, D.: Dimensions and principles of declassification. In: Proceedings CSFW’05. pp. 255–269. IEEE Computer Society (2005)
19. Sutherland, D.: A model of information. In: Proc. 9th National Computer Security Conference. pp. 175–183. DTIC Document (1986)
20. Zdancewic, S., Myers, A.C.: Observational determinism for concurrent program security. In: Proceedings CSFW’03 (2003)