The model checking problem for propositional intuitionistic logic with one variable is AC^1-complete

Martin Mundhenk and Felix Weiß

Institut für Informatik, Universität Jena
Jena, Germany
{martin.mundhenk,felix.weiss}@uni-jena.de

Abstract

We investigate the complexity of the model checking problem for propositional intuitionistic logic. We show that the model checking problem for intuitionistic logic with one variable is complete for logspace-uniform AC^1, and for intuitionistic logic with two variables it is P-complete. For superintuitionistic logics with one variable, we obtain NC^1-completeness for the model checking problem and for the tautology problem.

1998 ACM Subject Classification F.2 Analysis of algorithms and problem complexity, F.4 Mathematical logic and formal languages

Keywords and phrases complexity, intuitionistic logic, model checking, AC^1

Digital Object Identifier 10.4230/LIPIcs.STACS.2011.368

1 Introduction

Intuitionistic logic (see e.g. [8, 20]) is a part of classical logic that can be proven using constructive proofs–e.g. by proofs that do not use *reductio ad absurdum*. For example, the law of the excluded middle $a \vee \neg a$ and the weak law of the excluded middle $\neg a \vee \neg\neg a$ do not have a constructive proof and are not valid in intuitionistic logic. Not surprisingly, constructivism has its costs. Whereas the tautology problem is coNP-complete for classical propositional logic [5], for intuitionistic propositional logic $\mathcal{IPC}$ it is PSPACE-complete [17, 18]. The computational hardness of intuitionistic logic is already reached with the fragment $\mathcal{IPC}_2$ of formulas having only two variables: the tautology problem is PSPACE-complete already for $\mathcal{IPC}_2$ [16]. Recall that every fragment of classical propositional logic with a fixed number of variables has an NC^1-complete tautology problem (follows from [2]).

In this paper, we consider the complexity of intuitionistic propositional logic with one or two variables. The model checking problem—i.e. the problem to determine whether a given formula is satisfied by a given Kripke model—was recently shown to be P-complete [12] for $\mathcal{IPC}$. We show, that it remains P-complete for the fragment with two variables $\mathcal{IPC}_2$. More surprisingly, for the fragment with one variable $\mathcal{IPC}_1$ we show the model checking problem to be AC^1-complete. A basic ingredient for this result lies in normal forms for models and formulas as found by Nishimura [14], that we reinvestigate under an algorithmic and complexity theoretical point of view. To our knowledge, this is the first "natural" AC^1-complete problem, whereas formerly known AC^1-complete problems (see e.g. [1]) have some explicit logarithmic bound in the problem definition. In contrast, the formula value problem for classical propositional logic is NC^1-complete [2], even with one variable (follows from [2]).

28th Symposium on Theoretical Aspects of Computer Science (STACS'11).
Editors: Thomas Schwentick, Christoph Dürr; pp. 368–379

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany


Classical propositional logic is the extension of $\mathcal{IPC}$ with the axiom $a \vee \neg a$. Those proper extensions of intuitionistic logic are called superintuitionistic logics. The superintuitionistic logic $\mathcal{KC}$ (see [7]) results from adding $\neg a \vee \neg\neg a$ to $\mathcal{IPC}$. We show that the model checking problem for every superintuitionistic logic with one variable is NC^1-complete (and easier than that for $\mathcal{IPC}_1$), whereas for the superintuitionistic logic $\mathcal{KC}$ with two variables it is already P-complete (and as hard as for $\mathcal{IPC}_2$).

We also consider the tautology problem that is known to be PSPACE-complete for $\mathcal{IPC}_2$ [16]. Svejdar [19] recently showed the upper bound $\mathsf{SPACE}(\log n \cdot \log\log n)$ for $\mathcal{IPC}_1$. We show the tautology problem to be in NC^1 for any superintuitionistic logic with one variable. For superintuitionistic logics with more than one variable such a general result for the tautology problem remains open.

This paper is organized as follows. In Section 2 we introduce the notations we use for intuitionistic logic and model checking. Section 3 is devoted to introduce the old results by Nishimura [14] and to upgrade them with a complexity analysis. The following Section 4 presents our lower and upper bound for model checking for $\mathcal{IPC}_1$. Section 5 deals with the complexity of the model checking problem and the tautology problem for superintuitionistic logics with one variable, and Section 6 considers the case with two variables. The implied completeness for the model checking for intuitionistic logic and conclusions are drawn in Section 7. Proofs and technical details can be found in [13].

2 Preliminaries

Complexity (see e. g. [21]). The notion of reducibility we use is the logspace many-one reducibility $\leq_{\mathrm{m}}^{\log}$, except for NC^1 hardness, where we use first-order reducibility. NC^1 and AC^1 are the classes of sets that are decided by families of logspace-uniform circuits of polynomial size and logarithmic depth. The circuits consist of and-, or-, and negation-gates. The negation-gates have fan-in 1. For NC^1, the and- and or-gates have fan-in 2 (bounded fan-in), whereas for AC^1 there is no bound on the fan-in of the gates (unbounded fan-in). $\mathsf{ALOGTIME}$ denotes the class of sets decided by alternating Turing machines in logarithmic time, and we will use that $\mathsf{NC}^1 = \mathsf{ALOGTIME}$ (see [15]). L denotes the class of sets decidable in logarithmic space. We use $\mathsf{ALOGSPACE}[f(n)]$ to denote the class of sets decided by an alternating log-space Turing machine that makes $O(f(n))$ alternations, where n is the length of the input. We will use that $\mathsf{AC}^1 = \mathsf{ALOGSPACE}[\log n]$ (see [6]). $\mathsf{LOGDCFL}$ is the class of sets that are $\leq_{\mathrm{m}}^{\log}$-reducible to deterministic context-free languages. It is also characterized as the class of sets decidable by deterministic Turing machines in polynomial-time and logarithmic space with additional use of a stack. The inclusion structure of the classes we use is as follows.

$$\mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{LOGDCFL} \subseteq \mathsf{AC}^1 \subseteq \mathsf{P} \subseteq \mathsf{PSPACE}$$

Intuitionistic Propositional Logic (see e.g.[20]). Let VAR denote a countable set of *variables*. The language $\mathcal{IPC}$ of intuitionistic propositional logic is the same as that of propositional logic $\mathcal{PC}$, i.e. it is the set of all formulas of the form

$$\varphi ::= p \mid \bot \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi)\,,$$

where $p \in \mathrm{VAR}$. The languages $\mathcal{IPC}_i$ are the subsets/fragments of $\mathcal{IPC}$ for which VAR consists of i variables. We will consider $\mathcal{IPC}_0$ (i.e. formulas without variables), $\mathcal{IPC}_1$ and $\mathcal{IPC}_2$ (i.e. formulas with one resp. two variables).

As usual, we use the abbreviations $\neg\varphi := \varphi \to \bot$ and $\top := \neg\bot$. Because of the semantics of intuitionistic logic, one cannot express $\wedge$ or $\vee$ using $\to$ and $\bot$.

A *Kripke model* for intuitionistic logic is a triple $\mathcal{M} = (U, R, \xi)$, where U is a nonempty and finite set of *states*, R is a preorder on U (i.e. a reflexive and transitive binary relation), and $\xi : \mathrm{VAR} \to \mathfrak{P}(U)$ is a function — the *valuation function*. Informally speaking, for any variable it assigns the set of states in which this variable is satisfied. The valuation function ξ is monotone in the sense that for every $p \in \mathrm{VAR}$, $a, b \in U$: if $a \in \xi(p)$ and aRb, then $b \in \xi(p)$. (U, R) can also be seen as a directed graph. We will call such models *intuitionistic*.

Given an intuitionistic model $\mathcal{M} = (U, \leqslant, \xi)$ and a state $s \in U$, the *satisfaction relation for intuitionistic logics* $\models$ is defined as follows.

$$
\begin{aligned}
&\mathcal{M}, s \not\models \bot \\
&\mathcal{M}, s \models p && \text{iff } \ s \in \xi(p),\ p \in \mathrm{VAR}, \\
&\mathcal{M}, s \models \varphi \wedge \psi && \text{iff } \ \mathcal{M}, s \models \varphi \text{ and } \mathcal{M}, s \models \psi, \\
&\mathcal{M}, s \models \varphi \vee \psi && \text{iff } \ \mathcal{M}, s \models \varphi \text{ or } \mathcal{M}, s \models \psi, \\
&\mathcal{M}, s \models \varphi \to \psi && \text{iff } \ \forall n \geqslant s : \text{ if } \mathcal{M}, n \models \varphi \text{ then } \mathcal{M}, n \models \psi
\end{aligned}
$$

A formula φ is *satisfied* by an intuitionistic model $\mathcal{M}$ in state s iff $\mathcal{M}, s \models \varphi$. A *tautology* resp. a *valid formula* is a formula that is satisfied by every model.

The Model Checking Problem. This paper examines the complexity of model checking problems for intuitionistic logics.

Problem:	model checking problem for $\mathcal{IPC}_i$
Input:	$\langle \varphi, \mathcal{M}, s \rangle$, where φ is an $\mathcal{IPC}_i$ formula, $\mathcal{M}$ is an intuitionistic Kripke model, and s is a state of $\mathcal{M}$
Question:	$\mathcal{M}, s \models \varphi$?

We assume that formulas and Kripke models are encoded in a straightforward way. This means, a formula is given as a text, and the graph (U, R) of a Kripke model is given by its adjacency matrix that takes $|U|^2$ bits. Therefore, only finite Kripke models can be considered.

3 Properties of $\mathcal{IPC}_1$ and its complexity

The set $\mathcal{IPC}_1$ of formulas with one variable is partitioned into infinitely many equivalence classes [14]. This was shown using the formulas that are inductively defined as follows (see e.g.[8]). We use a for the only variable.

$$\varphi_1 = \neg a, \quad \psi_1 = a, \quad \varphi_{n+1} = \varphi_n \to \psi_n, \quad \psi_{n+1} = \varphi_n \vee \psi_n$$

The formulas $\bot, \top, \varphi_1, \psi_1, \varphi_2, \psi_2, \ldots$ are called *Rieger-Nishimura formulas*.

▶ **Theorem 1.** *([14], cf.[8, Chap.6,Thm.7]) Every formula in $\mathcal{IPC}_1$ is equivalent*[1] *to exactly one of the Rieger-Nishimura formulas.*

The function *RNindex* maps every formula to the index of its equivalent Rieger-Nishimura formula.

$$RNindex(\alpha) = \begin{cases} (i, phi), & \text{if} \quad \alpha \equiv \varphi_i \\ (i, psi), & \text{if} \quad \alpha \equiv \psi_i \\ (0, \bot), & \text{if} \quad \alpha \equiv \bot \\ (0, \top), & \text{if} \quad \alpha \equiv \top \end{cases}$$

[1] α is equivalent to β if every state in every model satisfies both or none formula.

We analyze the complexity of this function in Lemma 3. For this, we need a lower bound for the length of formulas in every equivalence class (see Lemma 2). With other words, this is an upper bound of the length of the Rieger-Nishimura index of any formula.

Let $[\varphi]$ denote the equivalence class that contains φ, for φ being an $\mathcal{IPC}_1$ formula. The equivalence classes of the $\mathcal{IPC}_1$ formulas form a free Heyting algebra over one generator (for algebraic details see [10]). This algebra is also called the *Rieger-Nishimura lattice* (see Fig. 1). It is shown in [14] that the lattice operations can be calculated using a big case distinction (see [13, Appendix A]). We use these algebraic properties of $\mathcal{IPC}_1$ to give a lower bound on the length of formulas[2] in the equivalence classes of $\mathcal{IPC}_1$, and to give an upper bound on the complexity of the problem to decide the Rieger-Nishimura index of a formula. Let $rank(\alpha)$ be the first element—the integer—of the $RNindex(\alpha)$ pair.

▶ **Lemma 2.** *For every $\mathcal{IPC}_1$ formula φ it holds that $rank(\varphi) \leq c_1 \cdot \log(|\varphi|)$, for a constant c_1 independent of φ.*

In order to analyze the complexity of the Rieger-Nishimura index computation, we define the following decision problem.

Problem:	EqRNformula
Input:	$\langle \alpha, (i,x) \rangle$, where α is an $\mathcal{IPC}_1$ formula and (i,x) is an index
Question:	Is (i,x) the Rieger-Nishimura index of α?

▶ **Lemma 3.** EqRNformula *is in* LOGDCFL.

Similar as any formula can be represented by its index, Kripke models can, too. We give a construction of models—the *canonical models*—(according to [8, Chap.6, Defi.5]) that are also used to distinguish the formula equivalence classes (Theorem 4). Our definition is a little different from that in [8, Chap.6, Defi.5]. We show in Lemma 5 that every model over one variable is homomorphic to a canonical model, and that this homomorphic Kripke model can be determined in ALOGSPACE$[n]$. (For model checking it suffices to determine at most $\log n$ as index. For details see Section 4.4.) For $n = 1, 2, \ldots$, we define the canonical models $\mathcal{H}_n = (W_n, \trianglelefteq, \xi_n)$ as follows (according to [8, Chap.6, Defi.5]).

1. $W_n = \{1, 2, \ldots, n-2\} \cup \{n\}$,
2. $a \trianglelefteq b$ iff $a = b$ or $a \geq b + 2$, and
3. $\xi_n(a) = \begin{cases} \emptyset, & \text{if } n = 2 \\ \{1\}, & \text{otherwise.} \end{cases}$

See Figure 1 for some examples.

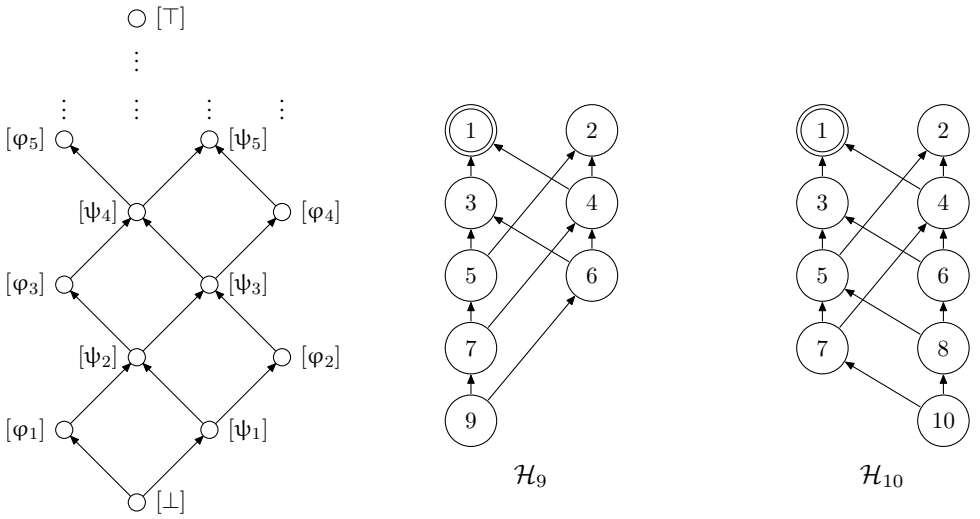
The formulas in $\mathcal{IPC}_1$ can be distinguished using the canonical models as follows.

▶ **Theorem 4.** *([14],cf.[8, Chap.6, Thm.8]) For every n and every k holds:*

1. $\mathcal{H}_n, n \models \psi_k$ *iff* $n \leq k$ *(i.e. $k \in \{n, n+1, \ldots\}$), and*
2. $\mathcal{H}_n, n \models \varphi_k$ *iff* $n < k$ *or* $n = k+1$ *(i.e. $k \in \{n-1\} \cup \{n+1, n+2, \ldots\}$).*

From [8, Chap.6, Lemma 11] it is known that every intuitionistic Kripke model $\mathcal{M}$ (for formulas with one variable a) is homomorphic to some $\mathcal{H}_i$. We additionally analyse the complexity of the decision problem whether $\mathcal{M}$ is homomorphic to $\mathcal{H}_i$ (see Lemma 5). For

[2] $|\alpha|$ denotes the length of the formula α, and it is the number of appearances of variables, connectives, and constants in α.



The Rieger-Nishimura lattice

$\mathcal{H}_9$

$\mathcal{H}_{10}$

Figure 1 The Rieger-Nishimura lattice (left), and the canonical models $\mathcal{H}_9$ and $\mathcal{H}_{10}$ (reflexive and transitive edges are not depicted, $\xi_n(a) = \{1\}$ is indicated by the double circle for state 1).

this, we define a function $\hbar$ that for given model $\mathcal{M}$ and state w has as function value $\hbar(\mathcal{M}, w)$ the index i of the homomorphic model $\mathcal{H}_i$. Let $\mathcal{M} = (W, \leqslant, \zeta)$ be a model and w a state. Let $W_{w\Uparrow} = \{v \in W \mid w \leqslant v\}$ and $W_{w\uparrow} = W_{w\Uparrow} \setminus \{w\}$. We define $\hbar$ as follows.

$$\hbar(\mathcal{M}, w) = \begin{cases} 1, & \text{if } w \in \zeta(a) \\ 2, & \text{if } w \notin \zeta(a) \text{ and } \forall v \in W_{w\uparrow}\ v \notin \zeta(a) \\ 3, & \text{if } w \notin \zeta(a) \text{ and } \forall v \in W_{w\uparrow}\ \hbar(\mathcal{M}, v) \neq 2 \text{ and } \exists u \in W_{w\uparrow}\ \hbar(\mathcal{M}, u) = 1 \\ n+2, & \text{if } \forall v \in W_{w\uparrow}\ \hbar(\mathcal{M}, v) \neq n+1 \text{ and} \\ & \quad \exists u_1, u_2 \in W_{w\uparrow}\ \hbar(\mathcal{M}, u_1) = n \text{ and } \hbar(\mathcal{M}, u_2) = n-1 \end{cases}$$

We call $\hbar(\mathcal{M}, w)$ the *model index* of w. The function $\hbar$ is well defined because for every state w it holds that $\{\hbar(\mathcal{M}, v) \mid v \in W_{w\Uparrow}\} = \{1, 2, \ldots, \hbar(\mathcal{M}, w) - 2\} \cup \{\hbar(\mathcal{M}, w)\}$.

Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be models, w_1 resp. w_2 a state from $\mathcal{M}_1$ resp. $\mathcal{M}_2$. We say that $(\mathcal{M}_1, w_1)$ is *homomorphic* to $(\mathcal{M}_2, w_2)$ if for every $\alpha \in \mathcal{IPC}_1$ it holds that $\mathcal{M}_1, w_1 \models \alpha$ iff $\mathcal{M}_2, w_2 \models \alpha$.

▶ **Lemma 5.** *Let $\mathcal{M} = (W, \leq, \zeta)$ be a Kripke model.*

1. *$(\mathcal{M}, w)$ is homomorphic to $(\mathcal{H}_{\hbar(\mathcal{M},w)}, \hbar(\mathcal{M}, w))$.*
2. *For a given model $\mathcal{M}$, a state w and an integer n, to decide whether $\hbar(\mathcal{M}, w) = n$ is in* ALOGSPACE$[n]$.

4 The complexity of model checking for $\mathcal{IPC}_1$

We first define an AC^1-hard graph problem, that is similar to the alternating graph accessibility problem, but has some additional simplicity properties. Then we give a construction that transforms such a graph into an intuitionistic Kripke model. This transformation is the basis for the reduction from the alternating graph accessibility problem to the model checking problem for $\mathcal{IPC}_1$.

4.1 Alternating graph problems

The alternating graph accessibility problem is shown to be P-complete in [4]. We use the following restricted version of this problem that is very similar to Boolean circuits with and- and or-gates (and input-gates). An *alternating slice graph* [12] $G = (V, E)$ is a directed bipartite acyclic graph with a bipartitioning $V = V_\exists \cup V_\forall$, and a further partitioning $V = V_0 \cup V_1 \cup V_2 \cup \cdots \cup V_{m-1}$ (m *slices*, $V_i \cap V_j = \emptyset$ if $i \neq j$) where $V_\exists = \bigcup_{i<m, i \text{ odd}} V_i$ and $V_\forall = \bigcup_{i<m, i \text{ even}} V_i$, such that $E \subseteq \bigcup_{i=1,2,\ldots,m-1} V_i \times V_{i-1}$ — i.e. all edges go from slice V_i to slice V_{i-1} (for $i = 1, 2, \ldots, m-1$). All nodes excepted those in the last slice V_0 have a positive outdegree. Nodes in $V_\exists$ are called *existential* nodes, and nodes in $V_\forall$ are called *universal* nodes. Alternating paths from node x to node y are defined as follows by the property $apath_G(x, y)$.

1) $apath_G(x, x)$ holds for all $x \in V$

2a) for $x \in V_\exists$: $apath_G(x, y)$ iff $\exists z \in V_\forall : (x, z) \in E$ and $apath_G(z, y)$

2b) for $x \in V_\forall$: $apath_G(x, y)$ iff $\forall z \in V_\exists :$ if $(x, z) \in E$ then $apath_G(z, y)$

The problem AsAgap is similar to the alternating graph accessibility problem, but for the restricted class of alternating slice graphs.

Problem:	AsAgap
Input:	$\langle G, s, t\rangle$, where $G = (V_\exists \cup V_\forall, E)$ is an alternating slice graph with slices $V_0, V_1, \ldots, V_{m-1}$, and $s \in V_{m-1} \cap V_\exists$, $t \in V_0 \cap V_\forall$
Question:	does $apath_G(s, t)$ hold?

Similarly as the alternating graph accessibility problem, AsAgap is P-complete [12, Lemma 2]. The following technical Lemma is not hard to prove.

▶ **Lemma 6.** *For every set A in (logspace-uniform)* AC^1 *exists a function f that maps instances x of A to instances $f(x) = \langle G_x, s_x, t_x\rangle$ of* AsAgap *and satisfies the following properties.*

1. *f is computable in logspace.*
2. *G_x is an alternating slice graph of logarithmic depth; i.e. if G_x has n nodes, then it has $m \leq \log n$ slices.*
3. *For all instances x of A holds: $x \in A$ iff $f(x) \in$* AsAgap*.*

Essentially, the function f constructs the AC^1 circuit $C_{|x|}$ with input x, and transforms it to an alternating slice graph G_x. The goal node t_x represents exactly the bits of x that are 1. The start node s_x corresponds to the output gate of $C_{|x|}$, and $apath_{G_x}(s_x, t_x)$ expresses that $C_{|x|}$ accepts input x.

4.2 Transforming alternating slice graphs to intuitionistic Kripke models

Our hardness results rely on a transformation of instances $\langle G, s, t\rangle$ of AsAgap to Kripke models $\mathcal{M}_G = (U, R, \xi)$. We describe it informally here. An example can be seen in Figure 2.

Let $\langle G, s, t\rangle$ be an instance of AsAgap for the slice graph $G = (V_\exists \cup V_\forall, E_G)$ with the m slices $V_\exists = V_{m-1} \cup V_{m-3} \cup \cdots \cup V_1$ and $V_\forall = V_{m-2} \cup V_{m-4} \cup \cdots \cup V_0$. The Kripke model $\mathcal{M}_G$ is constructed as follows. The nodes of the Kripke model consist of the nodes of $\mathcal{H}_{4m}$, and of two copies u^{in} and u^{out} of each node u of G. The slice $\mathcal{W}_i^{out}$ of $\mathcal{M}_G$ consists of the *out*-copies of nodes in V_i and the nodes $4i+1$ and $4i+2$ of $\mathcal{H}_{4m}$, and the slice $\mathcal{W}_i^{in}$ of $\mathcal{M}_G$ consists of the *in*-copies of nodes in V_i and the nodes $4i+3$ and $4i+4$ of $\mathcal{H}_{4m}$.

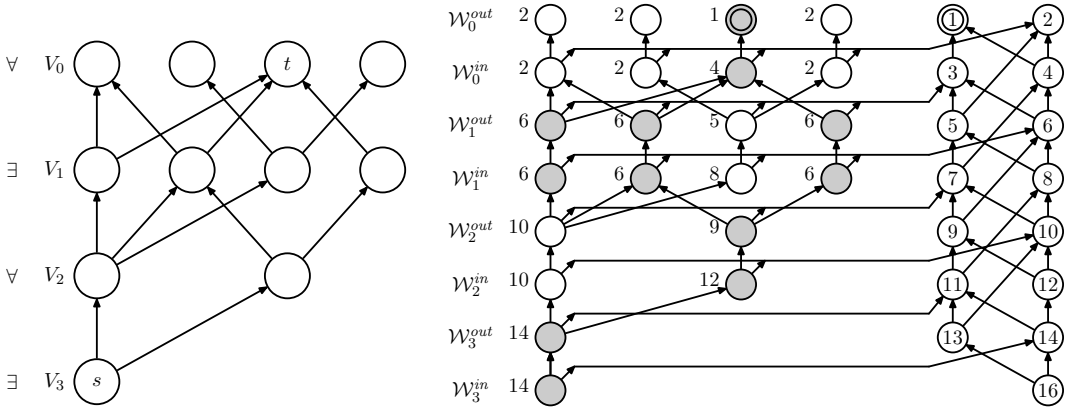


Figure 2 An alternating slice graph G (left) and the resulting Kripke model $\mathcal{M}_G$ (right); both the states in $\xi(a)$ are drawn doubly; pseudotransitive and reflexive edges in $\mathcal{M}_G$ are not depicted. The value at state x denotes its model index $\hbar(\mathcal{M}_G, x)$. For states in $\mathcal{H}_{16}$, their names and their model indices coincide. States v^{in} and v^{out} for which $apath_G(v,t)$ holds in G are colored grey.

All the edges of $\mathcal{H}_{4m}$ are kept. The edges (u,v) of G are changed to (u^{out}, v^{in}) in $\mathcal{M}_G$, and for every u of G an edge (u^{in}, u^{out}) is added. We add edges from the copies of the nodes in G to the nodes from $\mathcal{H}_{4m}$ as follows. Every node v^{out} for $v \in V_i$ $(i > 0)$ has an edge to node $4i-1$ from $\mathcal{H}_{4m}$, and every node v^{in} for $v \in V_i$ has an edge to node $4i+2$ from $\mathcal{H}_{4m}$.

An intuitionistic Kripke model must be transitive and reflexive. The part of the model that comes from $\mathcal{H}_{4m}$ is transitive and reflexive, the part of the model that comes from the alternating slice graph G is not. The transformation of the alternating slice graph to an intuitionistic Kripke model must be computable in logarithmic space, because it will be used as part of a logspace reduction function. Within this space bound we cannot compute the transitive closure of a graph. Therefore, we make the graph transitive with brute force. We add all edges from nodes v^{in} and v^{out} $(v \in V)$ that jump over at least one slice—we call these edges *pseudotransitive*. Finally, we need to add all missing reflexive edges.

The valuation function for our model $\mathcal{M}_G$ is $\xi(a) = \{t^{out}, 1\}$, where t^{out} is the copy of the goal node t in $\mathcal{W}_0^{out}$, and $\{1\} = \xi_{4m}(a)$ is the node from $\mathcal{H}_{4m}$. This concludes the informal description of the Kripke model $\mathcal{M}_G$. An example of an AsAgap instance $\langle G, s, t\rangle$ and the Kripke model $\mathcal{M}_G$ constructed from it can be seen in Figure 2.

The canonical model is attached to the slice graph in order to obtain control over the model indices of the other states (w.r.t. the model $\mathcal{M}_G$). This is described by Propositions 7 and 8.

▶ Proposition 7. For every $i = 0, 1, 2, \ldots, m-1$ and every $v \in V_i$ holds

$$\hbar(\mathcal{M}_G, v^{out}) \in \{4i+1, 4i+2\} \quad \text{and} \quad \hbar(\mathcal{M}_G, v^{in}) \in \{4i+2, 4i+4\} \quad .$$

▶ Proposition 8. For every $i = 0, 1, 2, \ldots, m-1$ and every $v \in V_i$ holds:

1. if i is even (∀ slice):
   $apath_G(v,t)$ iff $\hbar(\mathcal{M}_G, v^{out}) = 4i+1$, and $apath_G(v,t)$ iff $\hbar(\mathcal{M}_G, v^{in}) = 4i+4$,
2. if i is odd (∃ slice):
   $apath_G(v,t)$ iff $\hbar(\mathcal{M}_G, v^{out}) = 4i+2$, and $apath_G(v,t)$ iff $\hbar(\mathcal{M}_G, v^{in}) = 4i+2$.

Let T denote the function that maps instances $x = \langle G, s, t\rangle$ of AsAgap to Kripke models $T(x) = \mathcal{M}_G$ as described above. The following properties of T are easy to verify.

▶ **Lemma 9.** *1. T is logspace computable.*
2. If $x = \langle G, s, t\rangle$ for an alternating slice graph G with n nodes and $m < n$ slices, then $T(x)$ is a Kripke model with $\leq 4n$ states and depth $2m$.

We will use T as part of the reduction functions for our hardness results.

4.3 Hardness results

Our first result states that the calculation of the model index of an intuitionistic Kripke model is P-complete. It is already P-complete to decide the last bit of this model index.

▶ **Theorem 10.** *The following problems are* P*-complete.*
1. Given a Kripke model $\mathcal{M}$ and a state w, decide whether $\hbar(\mathcal{M}, w)$ is even.
2. Given a Kripke model $\mathcal{M}$, a state w, and an integer i, decide whether $\hbar(\mathcal{M}, w) = i$.

Proof. In order to show the P-hardness of the problems, we give a reduction from the P-hard problem AsAgap [12]. From an instance $\langle G, s, t\rangle$ of AsAgap where G is an alternating slice graph with m slices, construct $\mathcal{M} = T(\langle G, s, t\rangle)$. Then $\hbar(\mathcal{M}, s^{out}) \in \{4m+1, 4m+2\}$ (Proposition 7), and $apath_G(s, t)$ iff $\hbar(\mathcal{M}, s^{out}) = 4m{+}2$ (Proposition 8). Therefore, $\langle G, s, t\rangle \in$ AsAgap if and only if $\hbar(\mathcal{M}, s^{out})$ is even resp. $\hbar(\mathcal{M}, s^{out}) = 4m + 2$.

For every model $\mathcal{M} = (U, \leqslant, \xi)$ holds $\hbar(\mathcal{M}, w) \leq |U| + 1$. From Lemma 5 and using that $\mathsf{P} = \mathsf{ALOGSPACE}[poly]$ it then follows that both problems are in P. ◀

In the construction of the above proof, the decision whether $\hbar(\mathcal{M}, s^{out}) = 4m + 2$ is the same as to decide whether $\mathcal{M}, s^{out} \models \psi_{4m+2}$, for the Rieger-Nishimura formula ψ_{4m+2}. Unfortunately, the length of ψ_{4m+2} is exponential in m, and therefore the mapping from $\langle G, s, t\rangle$ (with m slices) to the model checking instance $\langle \psi_{4m+2}, T(\langle G, s, t\rangle), s^{out}\rangle$ cannot in general be performed in logarithmic space. But if the depth m of the slice graph is logarithmic, the respective formula ψ_{4m+2} has polynomial size only. Using Lemma 6, every AC^1 problem is logspace reducible to instances of AsAgap having logarithmically bounded depth, and by the above mapping these special instances are logspace reducible to model checking for $\mathcal{IPC}_1$.

▶ **Theorem 11.** *The model checking problem for $\mathcal{IPC}_1$ is AC^1-hard.*

4.4 Model checking for $\mathcal{IPC}_1$ is in AC^1

Algorithm 1 decides the model checking problem for $\mathcal{IPC}_1$. In the following, we estimate its complexity. The algorithm gets input $\langle \varphi, \mathcal{M}, s\rangle$ and works in two steps. In the first step we calculate the Rieger-Nishimura index (r, x) of φ. Since the index is very small (Lemma 2) and using Lemma 3, this can be done in LOGDCFL. In the second step we identify the homomorphic canonical model for $(\mathcal{M}, s)$. In fact it is not always necessary to identify the homomorphic canonical model exactly. According to Theorem 4, the input can be rejected if $\hbar(\mathcal{M}, s) > r + 1$, and the latter can be checked by finding some state $u \geqslant s$ in $\mathcal{M}$ with $\hbar(\mathcal{M}, u) > r + 1$. This estimation can be done in alternating logarithmic space with $r + 1$ alternations. If $\hbar(\mathcal{M}, u) \leq r + 1$, according to Theorem 5 the model index can be computed exactly in $\mathsf{ALOGSPACE}[r + 1]$, and knowing the Rieger-Nishimura index of φ and the model index of $(\mathcal{M}, s)$, it can be decided whether $\mathcal{M}, s \models \varphi$ using Theorem 4. Since k is at most about $\log|\varphi|$ (Lemma 2), the computation of the model index can be done in alternating logspace with $\log|\langle \varphi, \mathcal{M}, s\rangle|$ alternations (Lemma 5). Since the Rieger-Nishimura index of a formula can be decided in LOGDCFL, and $\mathsf{LOGDCFL} \subseteq \mathsf{AC}^1 = \mathsf{ALOGSPACE}[\log n]$, we obtain the desired upper bound.

Algorithm 1 model checking algorithm for $\mathcal{IPC}_1$

Require: a formula φ, a model $\mathcal{M}$ and a state s
search for (r, x) with $0 \leq r \leq c_1 \cdot \log(|\varphi|)$ such that $\langle \varphi, (r, x) \rangle \in$ EqRNformula
if $(r, x) = (0, \bot)$ **then** reject
else if $(r, x) = (0, \top)$ **then** accept
else if $x = psi$ **then**
  if $\hbar(\mathcal{M}, s) \neq i$ for all $i \in \{1, 2, \ldots, r\}$ **then** reject
  else accept
else if $x = phi$ **then**
  if $\hbar(\mathcal{M}, s) \neq i$ for all $i \in \{1, 2, \ldots, r-1\} \cup \{r+1\}$ **then** reject
  else accept
end if

▶ **Theorem 12.** *The model checking problem for $\mathcal{IPC}_1$ is in* AC^1.

5 Some notes on superintuitionistic logics with one variable

Superintuitionistic propositional logics are logics that have more valid formulas than $\mathcal{IPC}$. In this sense, classical propositional logic is a superintuitionistic logic, since it can be obtained as the closure under substitution and modus ponens of the tautologies from $\mathcal{IPC}$ plus $a \vee \neg a$ as additional axiom. A well-studied superintuitionistic logic is $\mathcal{KC}$ [7] that results from adding the weak law of the excluded middle $\neg a \vee \neg\neg a$ to $\mathcal{IPC}$. Semantically, the Kripke models for $\mathcal{KC}$ are restricted to those intuitionistic models $\mathcal{M} = (W, \leqslant, \xi)$ where $\leqslant$ is a *directed* preorder. Whereas $\mathcal{IPC}_1$ has infinitely many equivalence classes of formulas, $\mathcal{KC}_1$ has only 7 equivalence classes—represented by the Rieger-Nishimura formulas $\bot, \top, \varphi_1, \psi_1, \varphi_2, \psi_2, \psi_3$—that can be distinguished using the first 3 canonical models [14, 11]. The function $\hbar$ can be implemented as an alternating Turing machine that runs in logarithmic time, if the function value is fixed to a finite range—that in this case is $\{1, 2, 3\}$–independent on the input. For $\mathcal{KC}_1$, the Rieger-Nishimura index of the formulas also has a finite range (as mentioned above). Therefore, it can be calculated by an alternating Turing machine that runs in logarithmic time similar to the machine presented by Buss [3] that calculates the value of a Boolean formula. Instead of the Boolean values 0 and 1, for $\mathcal{KC}_1$ we have 7 different Rieger-Nishimura indices. The rules how the index of a formula can be calculated from the indices of its subformulas and the connective, follow directly from the Rieger-Nishimura lattice operations—see e.g. [13]. If the indices are bound to a finite range, this big table yields an even bigger but finite table without variable formula indices. For example, the equivalence $\varphi_n \vee \varphi_{n+1} \equiv \psi_{n+2}$ for all $n \geq 1$ induces the three equivalences $\varphi_1 \vee \varphi_2 \equiv \psi_3$, $\varphi_2 \vee \top \equiv \top$, and $\top \vee \top \equiv \top$ for $\mathcal{KC}_1$. This yields ALOGTIME as an upper bound for the tautology problem for $\mathcal{KC}_1$.

There are infinitely many superintuitionistic logics (with one variable) that can be obtained by adding an arbitrary formula as axiom to $\mathcal{IPC}_1$, that is not valid for $\mathcal{IPC}_1$. For example, if we add a formula equivalent to φ_k, then the superintuitionistic logic obtained has finitely many equivalence classes represented by $\bot, \top, \varphi_1, \psi_1, \varphi_2, \psi_2, \ldots, \varphi_{k-1}, \psi_{k-1}$. With similar arguments as for $\mathcal{KC}_1$ we can conclude that the model checking problems for these logics all are in NC^1. Moreover, the formula value problem for Boolean formulas without variables is NC^1-hard [2]. Intuitionistic formulas without variables have the same values, if they are interpreted as classical Boolean formulas. This means, the semantics of $\rightarrow$ is the same for Boolean formulas and for intuitionistic formulas without variables. Therefore, the model checking problem for any superintuitionistic logic is NC^1-hard, too.

▶ **Theorem 13.** *The model checking problem for every superintuitionistic logic with one variable is* NC^1*-complete.*

The tautology problem for superintuitionistic logic has the same complexity, since in order to decide whether a formula with one variable is a tautology it suffices to know its Rieger-Nishimura index.

▶ **Theorem 14.** *The tautology problem for every superintuitionistic logic with one variable is* NC^1*-complete.*

The best upper space bound for the tautology problem for $\mathcal{IPC}_1$ is a little higher, namely $\mathsf{SPACE}(\log n \cdot \log\log n)$ [19].

6 The complexity of model checking for $\mathcal{IPC}_2$

▶ **Theorem 15.** *The model checking problems for $\mathcal{KC}_2$ and for $\mathcal{IPC}_2$ are* P*-hard.*

This can be proven with a reduction from the $\mathcal{IPC}$ model checking problem that is P-complete [12]. We give formulas with two variables—the *replacement formulas*—which simulate the variables in an arbitrary $\mathcal{IPC}$ formula. For an arbitrary model we need to simulate the assignment function. For this we use a special model where every replacement formula has a unique maximal refuting state. We combine the given model and the special model in a way that if a variable is not assigned to a state, this state is connected to the state from the special model that refutes the replacement formula which substitutes this variable. This construction is similar to the polynomial time reduction from the tautology problem for $\mathcal{IPC}$ to the tautology problem for $\mathcal{IPC}_2$ in [16]. The main difference is that our logspace reduction yields pseudotransitive models, whereas in [16] a polynomial time reduction is used that allows to compute transitive models.

7 Conclusion

We consider computational problems that appear with intuitionistic propositional logic with at most two variables. Our main theorem completely characterizes the complexity of model checking for intuitionistic logic.

▶ **Theorem 16.**
1. *The model checking problem for $\mathcal{IPC}_0$ is* NC^1*-complete.*
2. *The model checking problem for $\mathcal{IPC}_1$ is* AC^1*-complete.*
3. *The model checking problem for $\mathcal{IPC}_2$ is* P*-complete.*

Part (1) follows from the fact that an intuitionistic formula that contains constants $\bot$ and $\top$ but no variables can be evaluated like a Boolean formula, whose evaluation problem is NC^1-complete [2] independently of the number of variables. Part (2) follows from Theorems 11 and 12. It shows a difference between $\mathcal{IPC}_1$ and its modal companion $\mathcal{S}4$ with one variable, for which the model checking problem is P-complete [12]. Part (3) comes from Theorem 15 (hardness) and [12] (containment in P). Similarly as Rybakov [16] for the tautology problem, this shows that the model checking problem for $\mathcal{IPC}$ reaches its full complexity already with the use of two variables.

Next we summarize the results from Theorems 13 and 15 for superintuitionistic logics.

1. *The model checking problem for every superintuitionistic logic with one variable is* NC^1*-complete.*
2. *The model checking problem for $\mathcal{KC}_2$ is* P*-complete.*

There are superintuitionistic logics with two variables between Boolean logic and $\mathcal{KC}_2$ (see below). The exact complexity of the model checking problem for these logics is open. It is interesting to notice that the complexity results for $\mathcal{IPC}_2$ and for $\mathcal{KC}_2$ are the same. But if only one variable is allowed, the complexity of $\mathcal{IPC}_1$ is higher than that of $\mathcal{KC}_1$.

Intuitionistic logic with one variable turns out to be the most challenging case. There are infinitely many equivalence classes of formulas, and according to Lemma 2 the sequence of smallest representatives of these equivalence classes has an exponential growth with respect to the length of the formulas. Such a fast growing sequence seems to appear rarely in "natural" problems, and it is a key ingredient for the AC^1-completeness of the model checking problem. Intuitionistic logic with one variable is strongly related to free Heyting algebras with one generator. Since Heyting algebras are generalizations of Boolean algebras, it would be interesting to investigate whether the difference between NC^1 and AC^1 is related to that between Boolean algebras and Heyting algebras.

If we consider other problems related to Kripke models for $\mathcal{IPC}_1$ that are not "out braked" by a very fast growing part of the input, the complexity jumps up to P-completeness, as shown in Theorem 10. Model checking for $\mathcal{IPC}_1$ also gets P-hard if the instances $\langle \varphi, \mathcal{M}, s \rangle$ allow the formula φ to be represented as a graph. Let us call this the *g-model checking problem* for short. This is a consequence of the P-hardness of the monotone circuit evaluation problem [9], holds even for formulas without variables, and therefore it also holds for all superintuitionistic logics. If formulas are represented as graphs, the sequence of smallest representatives of the equivalence classes of $\mathcal{IPC}_1$ does not have exponential growth anymore. Moreover, the calculation of the Rieger-Nishimura index gets P-hard.

▶ **Theorem 17.** *The following problems are* P*-complete:*

1. *the g-model checking problem for* $\mathcal{IPC}_1$*,*
2. *the g-model checking problem for every superintuitionistic logic with one variable,*
3. *the tautology problem for* $\mathcal{IPC}_1$*, where the formula is represented as a graph, and*
4. *the tautology problem for every superintuitionistic logic with one variable, where the formula is represented as a graph.*

Parts (1) and (2) contrast the different upper bounds NC^1 and AC^1 (Theorem 13 resp. Theorem 16) for the standard encodings of formulas. Parts (3) and (4) contrast the complexity of the tautology problems for the logics under consideration, that have the following upper bounds.

1. *The tautology problem for every superintuitionistic logic with one variable is* NC^1*-complete.*
2. *The tautology problem for* $\mathcal{IPC}_1$ *is in* $\mathsf{LOGDCFL} \cap \mathsf{SPACE}(\log n \cdot \log\log n)$.
3. *The tautology problem for* $\mathcal{KC}_2$ *and for* $\mathcal{IPC}_2$ *is* PSPACE*-complete.*

Part (1) is Theorem 14, part (2) is from Svejdar [19] and Lemma 3, and part (3) follows directly from [16]. The exact complexity of the tautology problem for $\mathcal{IPC}_1$ is open. It is interesting to notice that superintuitionistic logics with one variable all have lower complexity than $\mathcal{IPC}_1$, whereas for superintuitionistic logics with two variables already $\mathcal{KC}_2$ reaches the same complexity as $\mathcal{IPC}_2$. A superintuitionistic logic between Boolean logic and $\mathcal{KC}$ is $\mathcal{LC}$ [7]. Non-trivial hardness results for $\mathcal{LC}_2$ are not known. It would be interesting to investigate exactly the complexity jump from $\mathcal{LC}_2$ to $\mathcal{KC}_2$.

Acknowledgements The authors thank Vitek Svejdar, Heribert Vollmer, Johannes Süpke, Robert Zeranski, and Thomas Schneider for helpful discussions.

References

1 M. Beaudry and P. McKenzie. Circuits, matrices, and nonassociative computation. *J. Comput. Syst. Sci.*, 50(3):441–455, 1995.

2 S. R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proc. 19th STOC*, pages 123–131. ACM Press, 1987.

3 S. R. Buss. Algorithms for Boolean formula evaluation and for tree contraction. In *Arithmetic, Proof Theory, and Computational Complexity*, pages 96–115. Oxford University Press, 1993.

4 A. K. Chandra, D. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28:114–133, 1981.

5 S. A. Cook. The complexity of theorem proving procedures. In *Proc. 3rd STOC*, pages 151–158. ACM Press, 1971.

6 S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

7 M. Dummett and E. Lemmon. Modal logics between S4 and S5. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 14(24):250–264, 1959.

8 D. M. Gabbay. *Semantical investigations in Heyting's intuitionistic logic*. D.Reidel, Dordrecht, Boston, London, 1981.

9 L. M. Goldschlager. The monotone and planar circuit value problems are log space complete for P. *SIGACT News*, 9(2):25–29, 1977.

10 P. T. Johnstone. *Stone spaces*. Cambridge University Press, Cambridge, 1982.

11 D. Makinson. There are infinitely many diodorean modal functions. *J. of Symbolic Logic*, 31(3):406–408, 1966.

12 M. Mundhenk and F. Weiß. The complexity of model checking for intuitionistic logics and their modal companions. In *Proc. of RP 2010*, volume 6227 of *LNCS*, pages 146–160. Springer, 2010.

13 M. Mundhenk and F. Weiß. The model checking problem for propositional intuitionistic logic with one variable is AC^1-complete. *ArXiv e-prints*, abs/1012.3828v1, 2010.

14 I. Nishimura. On formulas of one variable in intuitionistic propositional calculus. *J. of Symbolic Logic*, 25:327–331, 1960.

15 W. L. Ruzzo. On uniform circuit complexity. *Journal of Computer and Systems Sciences*, 21:365–383, 1981.

16 M. N. Rybakov. Complexity of intuitionistic and Visser's basic and formal logics in finitely many variables. In *Papers from the 6th conference on "Advances in Modal Logic"*, pages 393–411. College Publications, 2006.

17 R. Statman. Intuitionistic propositional logic is polynomial-space complete. *Theor. Comput. Sci.*, 9:67–72, 1979.

18 V. Svejdar. On the polynomial-space completeness of intuitionistic propositional logic. *Arch. Math. Log.*, 42(7):711–716, 2003.

19 V. Svejdar. The tautology problem for IPC_1 is in space $\log n \cdot \log\log n$, 2009. Personal communication.

20 D. van Dalen. *Logic and Structure*. Springer, Berlin, Heidelberg, 4th edition, 2004.

21 H. Vollmer. *Introduction to Circuit Complexity – A Uniform Approach*. Texts in Theoretical Computer Science. Springer Verlag, Berlin Heidelberg, 1999.