

Interpolation in Classical Propositional Logic

Patrick Koopmann

Vrije Universiteit Amsterdam, The Netherlands

Christoph Wernhard

University of Potsdam, Germany

Frank Wolter

University of Liverpool, UK

– Draft August 15, 2025 –

Abstract

We introduce Craig interpolation and related notions such as uniform interpolation, Beth definability, and theory decomposition in classical propositional logic. We present four approaches to computing interpolants: via quantifier elimination, from formulas in disjunctive normal form, and by extraction from resolution or tableau refutations. We close with a discussion of the size of interpolants and links to circuit complexity.

Contents

1	Introduction	2
2	Preliminaries	3
3	Craig Interpolants	4
4	Uniform Interpolants	6
5	Beth Definability	8
6	Generalisations and an Application to Theory Decomposition	9
7	Interpolants via Quantifier Elimination	11
8	Interpolants via DNF	12
9	Interpolants via Resolution	14
9.1	Uniform Interpolation Using Resolution	15
9.2	Craig Interpolation Using Resolution	17
10	Craig Interpolants via Tableaux	20
10.1	A Tableau Proof System	21
10.2	Craig-Lyndon Separator Extraction from a Closed Tableau	22

© 2025 Copyright for this paper by its authors

DRAFT

Final version to appear in Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*, chapter XXXCHAPTER, pages XXXPFROM–XXXPTO. Ubiquity Press, 2026. XXXDOI.

2 Interpolation in Classical Propositional Logic

11 The Size of Interpolants	26
12 Bibliographic and Historic Remarks	29
12.1 Interpolation via Resolution	29
12.2 Interpolation via Tableaux	30
Acknowledgments	30
References	30

1 Introduction

In this chapter, we introduce Craig interpolation in the context of classical propositional logic. We aim to cover the fundamental results and techniques in the field, while still being accessible to readers who have only very basic logic background. We focus on explaining the underpinning logical notions and algorithms. For a discussion of applications of Craig interpolation in areas such as verification, databases, knowledge representation, and philosophy we refer the reader to the respective chapters of this volume.

In detail, we start by introducing the notion of a Craig interpolant between formulas in an implication and the Craig interpolation property which states that a Craig interpolant exists if the implication is valid. We then introduce uniform interpolants, a strong form of Craig interpolants which do not depend on the right hand side of the implication and which can be regarded as a formalization of forgetting some propositional atoms in a formula. We then link the computation of Craig interpolants to the computation of explicit definitions and the Craig interpolation property to the (projective) Beth definability property. While conceptually, and in applications, definability and interpolation are equally important, because of their mutual reducibility it suffices to discuss one of the two notions in detail. We chose interpolation and next discuss generalizations of Craig interpolation to structured inputs containing more than two formulas and an application to the uniqueness of theory decompositions. Then we present, in the core of this chapter, four ways of computing Craig (and sometimes uniform) interpolants. We start by presenting a simple, but conceptually important, construction that introduces quantified propositional formulas, in which interpolants trivially exist, and then applies quantifier eliminations to obtain propositional logic interpolants. Next we show how interpolants are obtained from formulas in disjunctive normal form by dropping atoms. Our focus is, however, on the introduction of two slightly more sophisticated and also practical methods of computing interpolants, namely their construction from resolution and tableau refutations. In both cases we give rather detailed expositions, assuming no prior understanding of resolution or tableau techniques. We close by discussing the size of interpolants and linking it to open questions in circuit complexity, again assuming hardly any prior knowledge of complexity theory.

2 Preliminaries

Propositional formulas are defined as usual by the grammar

$$\varphi, \psi \quad := \quad p \quad | \quad \top \quad | \quad \perp \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \psi \quad | \quad \varphi \vee \psi,$$

where p ranges over a countably infinite set of *propositional variables* or *atoms*. We set $\varphi \rightarrow \psi := \neg\varphi \vee \psi$ and $\varphi \leftrightarrow \psi := (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.

A *model* v is a mapping from the set of atoms to $\{0, 1\}$. The *truth value* $v(\varphi) \in \{0, 1\}$ of a formula φ under v is defined as usual by induction using truth tables. We set $v \models \varphi$ and say that φ is *satisfied in* v if $v(\varphi) = 1$. A formula φ is *valid*, in symbols $\models \varphi$, if $v \models \varphi$ for all models v , and φ is *satisfiable* if $v \models \varphi$ for some model v . We say that φ *entails* ψ , in symbols $\varphi \models \psi$, if $v \models \psi$ follows from $v \models \varphi$, for all models v . Observe that $\varphi \models \psi$ iff $\models \varphi \rightarrow \psi$. Formulas φ and ψ are *logically equivalent*, in symbols $\varphi \equiv \psi$, if $\varphi \models \psi$ and $\psi \models \varphi$.

The set $\text{sub}(\varphi)$ of *subformulas* of a formula φ is defined by induction by setting $\text{sub}(p) = \{p\}$, $\text{sub}(\top) = \{\top\}$, $\text{sub}(\perp) = \{\perp\}$, $\text{sub}(\varphi \wedge \psi) = \{\varphi \wedge \psi\} \cup \text{sub}(\varphi) \cup \text{sub}(\psi)$ and $\text{sub}(\neg\varphi) = \{\neg\varphi\} \cup \text{sub}(\varphi)$. Define the size $|\varphi|$ of a propositional formula φ as the number of its distinct subformulas. This corresponds to representing formulas as directed acyclic graphs (dags) and is the standard measure used in circuit complexity theory. We also call $|\varphi|$ the *dag-size* of φ .

In contrast, if one takes into account the number of different occurrences of subformulas, one obtains the *tree-size* $s(\varphi)$ of φ , defined inductively by setting $s(p) = s(\top) = s(\perp) = 1$, $s(\varphi \wedge \psi) = s(\varphi) + s(\psi) + 1$, and $s(\neg\varphi) = s(\varphi) + 1$. Clearly, $s(\varphi) \geq |\varphi|$ and it is conjectured that there is a superpolynomial (even exponential) gap between dag-size and tree-size in the sense that there are no functions e mapping every propositional formula φ to a logically equivalent propositional formula $e(\varphi)$ and polynomial function f (that is, function of the form $f(n) = cn^d$ with c, d natural numbers) such that $s(e(\varphi)) \leq f(|\varphi|)$ for all φ . However, this conjecture remains open [42].

Central for interpolation is the notion of a signature: A *signature* σ is a finite set of atoms. The signature $\text{sig}(\varphi)$ of a formula φ is the set of atoms that occur in it. In other words, the signature of φ are the non-logical symbols that occur in φ , whereas logical symbols such as $\wedge$, $\vee$, $\neg$, $\perp$ and $\top$ are not part of the signature of a formula. If $\text{sig}(\varphi) \subseteq \sigma$, we call φ a σ -*formula*. Given two formulas φ and ψ , we call ψ a *conservative extension* of φ if $\psi \models \varphi$, $\text{sig}(\varphi) \subseteq \text{sig}(\psi)$, and for every $\text{sig}(\varphi)$ -formula χ , $\psi \models \chi$ implies $\varphi \models \chi$.

Our methods for computing interpolants rely on different normal forms which we introduce next. A *literal* ℓ is either an atom or the negation $\neg p$ of an atom p . The *complement* of a literal ℓ , written as $\bar{\ell}$, is $\neg p$ if ℓ is p , and p if ℓ is $\neg p$. A disjunction of literals $C = \ell_1 \vee \dots \vee \ell_n$ is called a *clause*, often written as $C = \{\ell_1, \dots, \ell_n\}$ and treated as set. As usual, the empty disjunction is identified with $\perp$. A conjunction of literals $C = \ell_1 \wedge \dots \wedge \ell_n$ is called a *conjunctive clause*. The empty conjunction is identified with $\top$.

A formula φ is in *conjunctive normal form* (CNF) if it is a conjunction of clauses; it is in *disjunctive normal form* (DNF) if it is a disjunction of conjunctive clauses. Every formula

4 Interpolation in Classical Propositional Logic

is logically equivalent to a formula in CNF and DNF. For convenience, we may sometimes identify formulas in CNF/DNF with the corresponding sets of (conjunctive) clauses.

A formula φ is in *negation normal form (NNF)* if it is built from literals, truth value constants, conjunction and disjunction. Formally, NNF formulas are defined by the grammar

$$\varphi, \psi \quad := \quad p \mid \neg p \mid \top \mid \perp \mid \varphi \wedge \psi \mid \varphi \vee \psi,$$

where p ranges over the set of propositional atoms. CNF and DNF formulas are both special cases of NNF formulas. While any propositional formula can be converted in linear time to an equivalent NNF formula with an at most linear size increase, the smallest CNF and DNF formulas that are equivalent to a given formula may be exponentially larger.¹ One can, however, construct in polynomial time for every propositional formula φ a formula φ' in CNF that is a conservative extension of φ , using structural transformations [3] such as the Tseitin encoding [50].

3 Craig Interpolants

In this section we give the main definitions of this chapter.

► **Definition 1** (Craig Interpolant). *Let φ, ψ be formulas. Then a formula χ is called a Craig interpolant for φ, ψ if*

- $\varphi \models \chi$ and $\chi \models \psi$; and
- $\text{sig}(\chi) \subseteq \text{sig}(\varphi) \cap \text{sig}(\psi)$.

Clearly, if a Craig interpolant for φ, ψ exists, then $\varphi \models \psi$, by transitivity of the entailment relation $\models$. The converse direction, stating that a Craig interpolant for φ, ψ exists whenever $\varphi \models \psi$, is called the *Craig interpolation property (CIP)*. It holds for propositional logic.

► **Theorem 2** (Craig Interpolation Property (CIP)). *In propositional logic, if $\varphi \models \psi$, then there exists a Craig interpolant for φ, ψ .*

We will present and discuss several proofs of this result in Sections 7–10. Here we continue with illustrating the definition by examples, basic observations, and variants of the definition.

► **Example 3.**

(i) Observe that $p \wedge q_1 \models q_2 \rightarrow p$. By CIP a Craig interpolant for $p \wedge q_1, q_2 \rightarrow p$ exists. Clearly p is such a Craig interpolant. Moreover, up to logical equivalence, p is the only Craig interpolant for $p \wedge q_1, q_2 \rightarrow p$. Indeed, up to logical equivalence, the only other candidates are $\perp$, $\top$, and $\neg p$, and none of these is a Craig interpolant for $p \wedge q_1, q_2 \rightarrow p$.

¹ Each formula in DNF equivalent to $\bigwedge_{1 \leq i \leq n} (p_i \vee q_i)$ has at least 2^n conjunctive clauses. Each formula in CNF equivalent to $\bigvee_{1 \leq i \leq n} (p_i \wedge q_i)$ has at least 2^n clauses.

(ii) We have $p \wedge \neg p \models q$ and $\perp$ is a Craig interpolant for $p \wedge \neg p, q$. Up to logical equivalence, $\perp$ is the only Craig interpolant. This example shows that without admitting at least one of $\top$ or $\perp \equiv \neg\top$ as a formula not using any atoms, Craig interpolation would fail for formulas in disjoint signatures.

(iii) Consider atoms $p_1, \dots, p_n$. Then $\bigwedge_{1 \leq i \leq n} (p_i \wedge \neg p_i) \models \bigvee_{1 \leq i \leq n} (p_i \vee \neg p_i)$ and any formula using only atoms from $p_1, \dots, p_n$ is a Craig interpolant.

(iv) Let $\varphi = p \wedge q \wedge r$ and $\psi = s \rightarrow (p \vee q)$. Then $\varphi \models \psi$ and the Craig interpolants for φ, ψ are, up to logical equivalence, the formulas $p \wedge q, p, q$, and $p \vee q$.

(v) If φ is a propositional formula that is semantically independent from some atoms $\sigma \subseteq \text{sig}(\varphi)$, i.e., φ is equivalent to a propositional formula ψ without atoms in σ , then Craig interpolation can be applied to find such a ψ . Specifically, any Craig interpolant for φ, φ' , where φ' is the formula obtained from φ by systematically replacing all atoms in σ with fresh atoms, provides such a formula ψ . For example, let

$$\varphi = (p \rightarrow (q \wedge r)) \wedge (p \vee q) \wedge (q \rightarrow r).$$

Let $\sigma = \{p\}$ and $\varphi' = (p' \rightarrow (q \wedge r)) \wedge (p' \vee q) \wedge (q \rightarrow r)$. We obtain $\psi = q \wedge r$, which is equivalent to φ , as Craig interpolant for φ, φ' . $\lrcorner$

Example 3.iii shows that the space of Craig interpolants for a fixed pair of formulas can be large. It has, however, a very transparent structure:

► **Theorem 4** (Closure of Craig Interpolants under Conjunction and Disjunction). *Let φ and ψ be formulas and $C(\varphi, \psi)$ be the set of Craig interpolants for φ, ψ . Then $C(\varphi, \psi)$ is closed under conjunctions and disjunctions: if $\chi_1, \chi_2 \in C(\varphi, \psi)$, then $\chi_1 \wedge \chi_2 \in C(\varphi, \psi)$ and $\chi_1 \vee \chi_2 \in C(\varphi, \psi)$.*

As $C(\varphi, \psi)$ contains only finitely many formulas up to logical equivalence, it contains a logically strongest formula (that entails all others), given by the conjunction of all formulas in $C(\varphi, \psi)$, and a logically weakest formula (that is entailed by all others), given by the disjunction of all formulas in $C(\varphi, \psi)$.

In many applications and proofs, a different, but equivalent, view of Craig interpolants and CIP is useful.

► **Definition 5** (Craig Separator). *Let φ and ψ be formulas. Then a formula χ is a Craig separator for φ, ψ if*

- $\varphi \models \chi$ and $\chi \wedge \psi \models \perp$; and
- $\text{sig}(\chi) \subseteq \text{sig}(\varphi) \cap \text{sig}(\psi)$.

Clearly, χ is a Craig separator for φ, ψ iff χ is a Craig interpolant for $\varphi, \neg\psi$. The CIP is then equivalent to the statement that for any formulas φ, ψ , if $\varphi \wedge \psi$ is not satisfiable, then there exists a Craig separator for φ and ψ .

In the literature on verification, a Craig separator is often termed *reverse Craig interpolant* or even just *Craig interpolant*.

6 Interpolation in Classical Propositional Logic

Craig interpolation can be strengthened to take also the *polarity* of atom occurrences into account, which is often called *Craig-Lyndon* interpolation. Let $\text{sig}^\pm(\varphi)$ denote the set of all pairs $\langle p, \text{pol} \rangle$ such that atom p occurs in formula φ with polarity $\text{pol} \in \{+, -\}$, where the polarity of an occurrence is $+$ (respectively, $-$) if it is in the scope of an even (respectively, odd) number of negation symbols. We can then define the notion of *Craig-Lyndon interpolant* as follows.

► **Definition 6** (Craig-Lyndon Interpolant). *Let φ, ψ be formulas. Then a formula χ is called a Craig-Lyndon interpolant for φ, ψ if*

- $\varphi \models \chi$ and $\chi \models \psi$; and
- $\text{sig}^\pm(\chi) \subseteq \text{sig}^\pm(\varphi) \cap \text{sig}^\pm(\psi)$.

► **Example 7.** Let $\varphi = (p \rightarrow q) \wedge (r \rightarrow (p \vee q)) \wedge t$ and let $\psi = (p \rightarrow (q \wedge t)) \wedge ((q \wedge s) \rightarrow t)$. Atom p occurs in φ with both polarities but in ψ only negatively. Atom q occurs in ψ with both polarities but in φ only positively. Atom t occurs just positively, in both φ and ψ . Hence, in a Craig-Lyndon interpolant for φ, ψ the atom p is allowed only negatively, and q, t are allowed only positively. Atoms r, s are not allowed at all, since each of them occurs just in one of φ or ψ . Indeed, $\chi = (p \rightarrow q) \wedge t$ is a Craig-Lyndon interpolant for φ, ψ . The formula $\chi' = (p \rightarrow (q \wedge t)) \wedge (q \rightarrow t)$ is another Craig interpolant for φ, ψ , but it is no Craig-Lyndon interpolant as it has a negative occurrence of q . $\lrcorner$

A Craig-Lyndon interpolant is sometimes also called a *Lyndon interpolant* or a Craig interpolant with the *Lyndon property*. In propositional logic, the following strengthening of Theorem 2 holds.

► **Theorem 8** (Craig-Lyndon Interpolation Property (LIP)). *In propositional logic, if $\varphi \models \psi$, then there exists a Craig-Lyndon interpolant for φ, ψ .*

Since a Craig-Lyndon interpolant is also a Craig interpolant, the LIP (Theorem 8) implies the CIP (Theorem 2). In fact, some of the proofs in Sections 7–10 actually show the LIP. The notion of *Craig separator* also has an analog where polarity is considered.

► **Definition 9** (Craig-Lyndon Separator). *Let φ and ψ be formulas. Then a formula χ is a Craig-Lyndon separator for φ, ψ if*

- $\varphi \models \chi$ and $\chi \wedge \psi \models \perp$; and
- $\text{sig}^\pm(\chi) \subseteq \text{sig}^\pm(\varphi) \cap \text{sig}^\pm(\neg\psi)$.

As before, a formula χ is a Craig-Lyndon separator for φ, ψ iff χ is a Craig-Lyndon interpolant for $\varphi, \neg\psi$.

4 Uniform Interpolants

In this section we introduce a special form of Craig interpolants called uniform interpolants. While Craig interpolants are defined in terms of a pair of input formulas φ, ψ with $\varphi \models \psi$, uniform interpolants are defined in terms of a single formula φ and a signature $\sigma \subseteq \text{sig}(\varphi)$.

► **Definition 10** (Uniform Interpolant). *Let φ be a propositional formula and $\sigma \subseteq \text{sig}(\varphi)$. Then a propositional formula χ is called a uniform σ -interpolant for φ if*

- $\varphi \models \chi$;
- $\text{sig}(\chi) \subseteq \sigma$; and
- *for every propositional formula ψ , if $\varphi \models \psi$ and $\text{sig}(\psi) \cap \text{sig}(\varphi) \subseteq \sigma$, then $\chi \models \psi$.*²

Observe that, given φ and $\sigma \subseteq \text{sig}(\varphi)$, a uniform σ -interpolant for φ is a Craig interpolant for all pairs φ, ψ such that $\varphi \models \psi$ and ψ shares with φ only atoms that are in σ . In contrast to Craig interpolants, uniform interpolants are uniquely determined (up to logical equivalence) by φ and σ . In what follows, we therefore refer to *the* uniform σ -interpolant for a formula φ .

► **Example 11.** We consider the uniform σ -interpolants for the formulas φ in the entailments $\varphi \models \psi$ in Example 3, where $\sigma = \text{sig}(\varphi) \cap \text{sig}(\psi)$. The uniform $\{q_1\}$ -interpolant for $p \wedge q_1$ is given by p and the uniform $\emptyset$ -interpolant for $p \wedge \neg p$ is given by $\perp$. Similarly, the uniform $\{p_1, \dots, p_n\}$ -interpolant for $\bigwedge_{1 \leq i \leq n} (p_i \wedge \neg p_i)$ is given by $\perp$. Finally, the uniform $\{p, q\}$ -interpolant for $p \wedge q \wedge r$ is $p \wedge q$. $\perp$

Note that in Example 11, the uniform σ -interpolant is always the logically strongest Craig interpolant for φ, ψ . This is no accident. To show this in general, assume that χ is the logically strongest Craig interpolant for φ, ψ . Let $\sigma = \text{sig}(\varphi) \cap \text{sig}(\psi)$ and let χ' be the uniform σ -interpolant for φ . We show that χ and χ' are logically equivalent. Clearly $\chi' \models \chi$ since $\varphi \models \chi$ and $\text{sig}(\chi) \subseteq \sigma$. We also have $\chi \models \chi'$ since χ' is a Craig interpolant for φ, ψ and χ is the logically strongest one.

Sometimes, uniform interpolants as in Definition 10 are also called *right-uniform interpolants*, indicating that it is a Craig interpolant for an implication $\varphi \models \psi$ that is independent of the formula *on the right*. This notion can then be contrasted with that of a *left-uniform interpolant*, which is a Craig interpolant that is independent of the formula on the left (it is a Craig interpolant also for any other entailment $\varphi' \models \psi$):

► **Definition 12** (Left-Uniform Interpolant). *Let φ be a propositional formula and $\sigma \subseteq \text{sig}(\varphi)$. Then a propositional formula χ is called a left-uniform σ -interpolant for φ if*

- $\chi \models \varphi$;
- $\text{sig}(\chi) \subseteq \sigma$; and
- *for every propositional formula ψ , if $\psi \models \varphi$ and $\text{sig}(\psi) \cap \text{sig}(\varphi) \subseteq \sigma$, then $\psi \models \chi$.*

For logics with classical negation, right-uniform interpolation can easily be reduced to left-uniform interpolation and vice versa: χ is a right-uniform σ -interpolant for ψ iff $\neg\chi$ is a left-uniform σ -interpolant for $\neg\psi$. For the rest of this chapter, we therefore focus on right-uniform interpolants and call them simply uniform interpolants.

² Uniform interpolants are not always defined in exactly the same way. For instance, instead of starting from the signature σ of the uniform interpolant, one can start from the signature $\text{sig}(\varphi) \setminus \sigma$ that is ‘forgotten’, or the condition $\text{sig}(\psi) \cap \text{sig}(\varphi) \subseteq \sigma$ is replaced by the condition $\text{sig}(\psi) \subseteq \sigma$. In the context of propositional logic these different definitions are equivalent.

8 Interpolation in Classical Propositional Logic

The following result is shown in different ways in Sections 7–9.

► **Theorem 13** (Uniform Interpolation Property (UIP)). *In propositional logic, if φ is a propositional formula and $\sigma \subseteq \text{sig}(\varphi)$, then a uniform σ -interpolant for φ exists.*

Note that any logic with the uniform interpolation property automatically satisfies the CIP.

5 Beth Definability

In this section, we show that Craig interpolants are closely related to definitions and can be used to compute an explicit definition of an atom that is implicitly defined. We first formalize the notions of explicit and implicit definitions.

► **Definition 14** (Explicit Definability). *Let φ be a propositional formula, σ be a set of atoms, and $p \notin \sigma$. Then a propositional formula ψ is a σ -definition of p under φ if $\text{sig}(\psi) \subseteq \sigma$ and $\varphi \models p \leftrightarrow \psi$. p is called explicitly σ -definable under φ if there exists a σ -definition of p under φ .*

The condition $p \notin \sigma$ reflects the intuition that definitions should not be circular. Also note that we are not stipulating a definition of p but ask whether p is logically equivalent to some ψ under the assumption φ .

Alternatively, we may approach definability from a semantic point of view: The atom p is *implicitly σ -definable under φ* if, whenever two models that satisfy φ agree on the values of all atoms in σ , then they also agree on the value of p . Intuitively, the interpretation of the atoms in σ fully determines the interpretation of p . The formal definition is as follows.

► **Definition 15** (Implicit Definability). *Let φ be a propositional formula, σ be a set of atoms, and $p \notin \sigma$. Then p is implicitly σ -definable under φ if for any models v_1, v_2 satisfying φ , $v_1(q) = v_2(q)$ for all $q \in \sigma$ implies $v_1(p) = v_2(p)$.*

Clearly explicit definability implies implicit definability. The converse is called *projective Beth definability property (BDP)* and does not hold for all logics. It does hold, however, for classical propositional logic.

► **Theorem 16** (Projective Beth Definability Property (BDP)). *Propositional logic has the projective Beth definability property.*

Proof. The proof is by reduction to the Craig interpolation property. Assume p is implicitly σ -definable under φ . Let φ' be obtained from φ by systematically replacing all atoms q not in σ by associated fresh atoms q' . Then implicit definability implies

$$\varphi \wedge \varphi' \models p \leftrightarrow p'$$

Hence

$$\varphi \wedge p \models \varphi' \rightarrow p'$$

Any Craig interpolant χ of $\varphi \wedge p, \varphi' \rightarrow p'$ is a σ -definition of p under φ . ◀

Observe that the proof of Theorem 16 shows how one can construct in polynomial time explicit definitions from Craig interpolants: given φ , a signature σ , and $p \notin \sigma$, a σ -definition of p under φ is obtained by taking any Craig interpolant for propositional formulas χ, χ' computed from φ , σ , and p in linear time.

The special case of the projective Beth definability property where σ is not a parameter but defined as $\sigma = \text{sig}(\varphi) \setminus \{p\}$ is known as the *Beth definability property*. Note also that the notions of explicit and implicit definability, and the projective Beth definability property can be easily generalized to definitions of compound propositional formulas in place of atoms p .

► **Example 17.**

(i) As a very simple first example for Beth definability consider $\varphi = p \leftrightarrow q$ and $\sigma = \{q\}$. Clearly p is σ -definable under φ . The σ -definition q is a Craig interpolant for $(\varphi \wedge p), (\varphi(p/p') \rightarrow p')$. The entailment underlying the Craig interpolation is $(p \leftrightarrow q) \wedge p \models (p' \leftrightarrow q) \rightarrow p'$, which can be equivalently expressed as $p \wedge q \models p' \vee q$.

(ii) Let $\varphi = (p \rightarrow (q \wedge r)) \wedge ((q \wedge s) \rightarrow p) \wedge (r \rightarrow s)$ and $\sigma = \{q, r\}$. Since $\varphi \models p \leftrightarrow (q \wedge r)$ atom p is σ -definable under φ . The σ -definition $q \wedge r$ is a Craig interpolant for $(\varphi \wedge p), (\varphi(p/p', s/s') \rightarrow p')$.

(iii) Let $\varphi = (p \rightarrow (q \wedge r)) \wedge ((q \vee r) \rightarrow p)$ and $\sigma = \{q, r\}$. Then p is σ -definable under φ , actually with several σ -definitions: $\varphi \models p \leftrightarrow (q \wedge r)$, $\varphi \models p \leftrightarrow q$, $\varphi \models p \leftrightarrow r$, and $\varphi \models p \leftrightarrow (q \vee r)$. Each of these four definitions is a Craig interpolant for $(\varphi \wedge p), (\varphi(p/p') \rightarrow p')$.

(iv) Let $\varphi = p$ and $\sigma = \{\}$. Then p is σ -definable under φ , since $p \models p \leftrightarrow \top$. The σ -definition $\top$ is a Craig interpolant of $(\varphi \wedge p), (\varphi(p/p') \rightarrow p')$, i.e., of $(p \wedge p), (p' \rightarrow p')$. $\square$

We also give a polynomial time reduction of the computation of Craig interpolants to the computation of explicit definitions.

► **Theorem 18** (Reduction of Craig Interpolation to Explicit Definitions). *Assume $\models \varphi \rightarrow \psi$. Then the following conditions are equivalent for any formula χ :*

- χ is a Craig interpolant for φ, ψ ;
- χ is an explicit $\text{sig}(\varphi) \cap \text{sig}(\psi)$ -definition of ψ under $\psi \rightarrow \varphi$.

Because of the mutual polynomial time reduction between Craig interpolant computation and explicit definition computation shown in this section, in what follows we focus on Craig interpolation without always mentioning again the consequences for explicit definitions.

6 Generalisations and an Application to Theory Decomposition

Craig interpolation has been generalised in many different ways. In particular, in applications of Craig interpolation to program verification various inductive generalisations have been introduced, including inductive sequences of interpolants [19, 36], tree interpolants [18, 7], and disjunctive interpolation [43, 44], in increasing order of generality, we refer the reader to [51] for further discussion. In all these generalisations, one considers multiple formulas (for

10 Interpolation in Classical Propositional Logic

instance, as labels of nodes in a tree), and the interpolants capture the interaction between these formulas in their shared signature.

Here we focus on a generalisation observed by Craig already [12, Lemma 2].

► **Theorem 19.** *In propositional logic, if $\varphi_1 \wedge \dots \wedge \varphi_k \models \varphi_{k+1} \vee \dots \vee \varphi_l$, then there are formulas $\chi_1, \dots, \chi_l$ with*

- $\varphi_i \models \chi_i$ for $1 \leq i \leq k$
- $\chi_i \models \varphi_i$ for $k+1 \leq i \leq l$
- $\text{sig}(\chi_i) \subseteq \text{sig}(\varphi_i) \cap \bigcup_{j \in \{1, \dots, l\}, j \neq i} \text{sig}(\varphi_j)$ for $1 \leq i \leq l$
- $\chi_1 \wedge \dots \wedge \chi_k \models \chi_{k+1} \vee \dots \vee \chi_l$

In applications to program verification, these interpolants have been studied for $l = k+1$ and $\varphi_{k+1} = \perp$ [34, Sect. 5], where they are called *symmetric interpolants*. We focus on a brief discussion of another special case of Theorem 19 called *parallel interpolation*. In what follows, T denotes a finite set of propositional formulas which, intuitively, axiomatises some theory or knowledge base of interest. We often identify T with the conjunction $\bigwedge_{\chi \in T} \chi$; for example, $T \models \varphi$ stands for $\bigwedge_{\chi \in T} \chi \models \varphi$. Now, parallel interpolation is the special case in which $l = k+1$ and $\text{sig}(\varphi_i) \cap \text{sig}(\varphi_j) = \emptyset$ for $1 \leq i < j \leq k$. So, given $T = \{\varphi_1, \dots, \varphi_n\}$ with $\text{sig}(\varphi_i) \cap \text{sig}(\varphi_j) = \emptyset$ for $1 \leq i < j \leq n$ and a formula ψ with $T \models \psi$, there are formulas $\chi_1, \dots, \chi_n$, called *parallel interpolants*, with

- $\varphi_i \models \chi_i$ for $1 \leq i \leq n$;
- $\text{sig}(\chi_i) \subseteq \text{sig}(\varphi_i) \cap \text{sig}(\psi)$ for $1 \leq i \leq n$; and
- $\{\chi_1, \dots, \chi_n\} \models \psi$.

Intuitively, the parallel interpolants χ_i capture, in the shared atoms of φ_i and ψ , the contribution of φ_i to the derivation of ψ from T . Although fairly straightforward to derive from Craig interpolation, parallel interpolation has some unexpected applications. Here we use it to show that finest decompositions of theories into axioms with mutually disjoint signatures are uniquely determined (up to logical equivalence). In detail, let T be a set of formulas and let $\sigma_1, \dots, \sigma_n$ be a partition of $\text{sig}(T)$. Then $\sigma_1, \dots, \sigma_n$ is called a *splitting* of T if there are formulas $\chi_1, \dots, \chi_n$ such that $\text{sig}(\chi_i) \subseteq \sigma_i$ for $1 \leq i \leq n$ and $\{\chi_1, \dots, \chi_n\}$ axiomatises T , that is to say, $\bigwedge_{1 \leq i \leq n} \chi_i$ and $\bigwedge_{\chi \in T} \chi$ are logically equivalent. Splittings formalise the idea of decomposing a theory T into axioms that speak about mutually disjoint subject matters. We say that a partition $\sigma_1, \dots, \sigma_n$ of $\text{sig}(T)$ is at least as fine as partition $\sigma'_1, \dots, \sigma'_m \subseteq \text{sig}(T)$ if for all σ'_i there exists σ_j with $\sigma_j \subseteq \sigma'_i$.

► **Theorem 20.** *Every finite set T of propositional formulas has a unique finest splitting.*

Proof. Assume splittings $\sigma_1, \dots, \sigma_n$ and $\sigma'_1, \dots, \sigma'_m$ of T are given. We show that the set of non-empty $\sigma_i \cap \sigma'_j$, with $1 \leq i \leq n$ and $1 \leq j \leq m$ are again a splitting of T . The claim then follows directly by assuming that $\sigma_1, \dots, \sigma_n$ and $\sigma'_1, \dots, \sigma'_m$ are finest splittings of T . Let $\varphi_1, \dots, \varphi_n$ and $\psi_1, \dots, \psi_m$ be axiomatisations of T witnessing that $\sigma_1, \dots, \sigma_n$ and $\sigma'_1, \dots, \sigma'_m$ are splittings of T , respectively. Then take the parallel interpolants χ_{ij} for the entailments

$\{\varphi_1, \dots, \varphi_n\} \models \psi_j$ with $\varphi_i \models \chi_{ij}$, for $1 \leq i \leq n$ and $1 \leq j \leq m$. Then $\text{sig}(\chi_{ij}) \subseteq \sigma_i \cap \sigma'_j$ and the χ_{ij} axiomatise T . So they witness that the non-empty $\sigma_i \cap \sigma'_j$ are a splitting of T . ◀

For further results on parallel interpolation and its application to theory decomposition in philosophy and knowledge representation we refer the reader to [29, 28, 38].

7 Interpolants via Quantifier Elimination

Having discussed various generalizations and applications of interpolation, in the next few sections, we discuss different proofs of the Craig interpolation property (CIP) and methods for computing interpolants. We start with the easiest proof, which shows the stronger result that propositional logic has the uniform interpolation property. CIP then follows (Theorem 13). While the proof constructs uniform interpolants (and so Craig interpolants), it does not show how to construct interpolants efficiently. We examine efficient methods in Sections 8–10.

The proof has two steps: (i) we show that uniform interpolants always exist in the language obtained from propositional logic by adding quantifiers over atoms; and (ii) we show that these quantifiers can be always eliminated in sense that one can always construct a logically equivalent propositional formula that does not use quantifiers. Overall, one then obtains that uniform interpolants always exist.

We first extend propositional logic by adding quantifiers. *QBF* (*quantified Boolean formulas*) are an extension of propositional logic defined according to the following grammar:

$$\varphi, \psi \quad := \quad p \quad | \quad \top \quad | \quad \perp \quad | \quad \neg\varphi \quad | \quad \varphi \wedge \psi \quad | \quad \varphi \vee \psi \quad | \quad \exists p.\varphi$$

where p ranges over a countably infinite set of atoms. Models v of QBF are defined in the same way as for propositional logic and we define $v \models \varphi$ inductively using the additional condition

- $v \models \exists p.\varphi$ if there is a model v' that coincides with v for all atoms except possibly p such that $v' \models \varphi$.

Validity, satisfiability, and logical equivalence are now defined in the obvious way. The definition of the *signature* $\text{sig}(\varphi)$ of a QBF φ extends inductively the definition for propositional logic with $\text{sig}(\exists p.\varphi) = \text{sig}(\varphi) \setminus \{p\}$.

We are now in a position to give the mentioned proof of uniform interpolation in two steps.

QBF Uniform Interpolants. Assume φ and a signature $\sigma \subseteq \text{sig}(\varphi)$ are given. Let $p_1, \dots, p_n$ be any ordering of the atoms in $\text{sig}(\varphi) \setminus \sigma$ and consider the QBF $\chi = \exists p_1 \dots \exists p_n.\varphi$. Then

- $\varphi \models \chi$ and $\text{sig}(\chi) = \sigma$;
- for every propositional formula ψ , if $\varphi \models \psi$ and $\text{sig}(\psi) \cap \text{sig}(\varphi) \subseteq \sigma$, then $\chi \models \psi$.

and so χ satisfies the conditions for a uniform σ -interpolant for φ , except that it is not a propositional formula.

12 Interpolation in Classical Propositional Logic

Quantifier Elimination. Let $\varphi = \exists p_1 \cdots \exists p_n. \psi$ be a QBF with ψ a propositional formula. Then φ is logically equivalent to a propositional formula. Indeed, let $S_n = \{\perp, \top\}^n$ be the set of all sequences of $\perp$ and $\top$ of length n . Then

$$\bigvee_{t_1, \dots, t_n \in S_n} \psi[p_1/t_1, \dots, p_n/t_n]$$

is logically equivalent to φ , where $\psi[p_1/t_1, \dots, p_n/t_n]$ is obtained from ψ by replacing p_i by t_i for $1 \leq i \leq n$. Using quantifier elimination, we can thus easily compute propositional uniform interpolants from QBF uniform interpolants.

Note that it can now also be shown by induction that every QBF is logically equivalent to a propositional formula.

► **Example 21.** Let $\varphi = \exists q. (p \rightarrow q) \wedge (q \rightarrow r)$. By quantifier elimination φ is equivalent to $((p \rightarrow \perp) \wedge (\perp \rightarrow r)) \vee ((p \rightarrow \top) \wedge (\top \rightarrow r))$, which simplifies to $p \rightarrow r$. $\sqcup$

We conclude this section with a few further observations. Let $\forall p. \varphi := \neg \exists p. \neg \varphi$. When constructing Craig interpolants for formulas φ, ψ , instead of $\exists p_1 \cdots \exists p_n. \varphi$ with $p_1, \dots, p_n$ any ordering of $\text{sig}(\varphi) \setminus \text{sig}(\psi)$, we could have also used $\forall q_1 \cdots \forall q_n. \psi$ with $q_1, \dots, q_n$ any ordering of $\text{sig}(\psi) \setminus \text{sig}(\varphi)$. The formula $\exists p_1 \cdots \exists p_n. \varphi$ is (up to equivalence) the logically strongest interpolant for φ, ψ (it entails all others) and the formula $\forall q_1 \cdots \forall q_n. \psi$ is (up to equivalence) the logically weakest interpolant for φ, ψ (it is entailed by all others).

Quantifier elimination generalizes deciding satisfiability, which can be seen as follows. For $\exists p_1 \cdots \exists p_n. \varphi$ where $\{p_1, \dots, p_n\} = \text{sig}(\varphi)$ quantifier elimination yields a propositional formula built just from logical symbols. Rewriting with equivalences $\neg \perp \equiv \top$, $\neg \top \equiv \perp$, $\top \wedge \psi \equiv \psi$, $\perp \wedge \psi \equiv \perp$, $\top \vee \psi \equiv \top$, and $\perp \vee \psi \equiv \psi$ yields $\top$ if φ is satisfiable and $\perp$ if φ is unsatisfiable.

8 Interpolants via DNF

The uniform σ -interpolant for a formula φ formalises a natural notion of forgetting the atoms in $\text{sig}(\varphi) \setminus \sigma$ from φ . This suggests that one can construct the uniform σ -interpolant by somehow dropping the atoms in $\text{sig}(\varphi) \setminus \sigma$ from φ in a syntactic sense. This is clearly not the case in general. For instance, for $\varphi = (p \vee q) \wedge \neg p$ and $\sigma = \{p\}$, we do not obtain the uniform σ -interpolant for φ by just dropping q : the result of dropping q is logically equivalent to $p \wedge \neg p$ and $\varphi \not\models p \wedge \neg p$. If, however, φ is in DNF, then we indeed obtain the uniform σ -interpolant by dropping the atoms in σ .

Assume φ is given in DNF, say

$$\varphi = \bigvee_{1 \leq i \leq n} \varphi_i, \quad \varphi_i = \bigwedge_{1 \leq j \leq n_i} \ell_{ij}$$

with ℓ_{ij} literals. Let $\sigma \subseteq \text{sig}(\varphi)$. We may assume that no conjunctive clause φ_i contains an atom and its negation. Then obtain $\varphi^{-\sigma}$ from φ by dropping all literals ℓ_{ij} not in σ from every φ_i .

► **Lemma 22.** *Let $p_1, \dots, p_n$ be an ordering of $\text{sig}(\varphi) \setminus \sigma$. Then $\varphi^{-\sigma}$ is logically equivalent to $\exists p_1 \dots \exists p_n. \varphi$.*

Proof. The lemma can be proven as follows with a syntactic argument. We can move the existential quantifiers in $\exists p_1 \dots \exists p_n. \varphi$ inward, preserving equivalence, until all quantifications are of the form $\exists p_i. p_i$ or $\exists p_i. \neg p_i$ (we assume that the conjunctive clauses of φ contain no duplicate literals). We can do this due to the following equivalences: 1) $\exists p_1 \dots \exists p_n. \varphi \vee \psi \equiv \exists p_1 \dots \exists p_n. \varphi \vee \exists p_1 \dots \exists p_n. \psi$; 2) if $p \notin \text{sig}(\psi)$, then $\exists p. \varphi \wedge \psi \equiv (\exists p. \varphi) \wedge \psi$; and 3) if $p \notin \text{sig}(\varphi)$, then $\exists p. \varphi \equiv \varphi$.

Then we apply our quantifier elimination method from Section 7 to these quantified literals. For positive as well as negative literals this yields $\top$, i.e., $\exists p_i. p_i \equiv p_i[p_i/\perp] \vee p_i[p_i/\top] \equiv \top$ and $\exists p_i. \neg p_i \equiv \neg p_i[p_i/\perp] \vee \neg p_i[p_i/\top] \equiv \top$. By removing these redundant subformulas $\top$ from the conjunctive clauses, we obtain exactly $\varphi^{-\sigma}$. ◀

We can generalize this idea of computing uniform interpolants to the computation of Craig interpolants, for which we take into account the logical strength of the right hand side of the entailment. It is more convenient to look at Craig separators here (the construction of Craig interpolants would assume the right-hand side to be in CNF rather than DNF). Assume that

$$\varphi = \bigvee_{1 \leq i \leq n} \varphi_i, \quad \varphi_i = \bigwedge_{1 \leq j \leq n_i} \ell_{ij}, \quad \psi = \bigvee_{1 \leq i' \leq m} \psi_{i'}, \quad \psi_{i'} = \bigwedge_{1 \leq j' \leq m_{i'}} \ell'_{i'j'}$$

and that $\varphi \wedge \psi$ is not satisfiable. We may assume that no φ_i nor $\psi_{i'}$ contains an atom and its negation. We construct a Craig separator as follows. Let $\sigma = \text{sig}(\varphi) \cap \text{sig}(\psi)$. Because $\varphi \wedge \psi$ is not satisfiable, for any pair i, i' we find j, j' s.t. $\ell_{ij} \wedge \ell'_{i'j'}$ unsatisfiable. Take any collection P of such pairs $(\ell_{ij}, \ell'_{i'j'})$. Then P can be regarded as a *proof* that $\varphi \wedge \psi$ is not satisfiable. Observe that each pair in P consists of an atom in σ and its negation. The formula φ_P now is the disjunction over all literals from φ that contribute to a pair in P . It is easy to check that φ_P satisfies all properties of a Craig separator: $\varphi \models \varphi_P$, $\text{sig}(\varphi_P) \subseteq \sigma$, and $\varphi_P \wedge \psi$ is unsatisfiable since the pairs in P still show that $\varphi_P \wedge \psi$ is unsatisfiable. Indeed, it is even a Craig-Lyndon separator: Any literal in φ_P occurs as a literal ℓ_{ij} in φ , while its complement occurs as a literal $\ell'_{i'j'}$ in ψ . We obtain the following theorem, which is at the same time our first proof of Theorem 8, namely that propositional logic has the LIP.

► **Theorem 23.** *φ_P is a Craig-Lyndon separator for φ, ψ .*

► **Example 24.** Let $\varphi = p \wedge q \wedge r$ and $\psi = s \rightarrow (p \vee q)$, as in Example 3.iv, and let $\sigma = \text{sig}(\varphi) \cap \text{sig}(\psi) = \{p, q\}$. Formula φ is already in DNF, a single conjunctive clause. As $\varphi^{-\sigma}$ we obtain $p \wedge q$ by dropping atom r from this conjunctive clause. The formula $p \wedge q$ is the unique uniform $\{p, q\}$ -interpolant for φ . Now consider $\psi' = s \wedge \neg p \wedge \neg q$, the negation of ψ in DNF, also a single conjunctive clause. A collection of pairs $(\ell_{ij}, \ell'_{i'j'})$ as described above is $\{(\ell_{11}, \ell'_{12})\} = \{(p, \neg p)\}$. From this pair, we obtain p as Craig separator for φ, ψ' (and as Craig interpolant for φ, ψ). Another suitable set of pairs is $\{(\ell_{12}, \ell'_{13})\} = \{(q, \neg q)\}$. Based on this we obtain q as another Craig separator. ┘

14 Interpolation in Classical Propositional Logic

9 Interpolants via Resolution

The quantifier elimination method described in Section 7 gives an elegant proof of the CIP, and a simple method for computing Craig and uniform interpolants. A downside however is that the interpolants computed in this way are formulas whose size is always exponential in the number of eliminated atoms. In contrast, the method described in Section 8 does not increase the size of the involved formula. However, it requires these formulas to be in DNF, and transforming arbitrary formulas into DNF can lead to an exponential increase in the size of the formula. Other methods for interpolation try to compute interpolants in a more goal-oriented manner, with the aim of achieving efficiency in many practical cases, even though so far no method is known that can compute interpolants whose size is guaranteed to be sub-exponential, and there might not even exist one (this is discussed in more detail in Section 11).

One such method uses resolution. Resolution is a well-known method for deciding satisfiability of propositional formulas in CNF [4, 26]. As mentioned in Section 2, it is possible to transform any propositional formula into a conservative extension in CNF in linear time. In particular, many structure-preserving transformations such as the well-known Tseitin encoding produce formulas that are not only equi-satisfiable, but also conservative extensions [3, 50], which thanks to the following result is sufficient for computing interpolants. Since we will focus on Craig separators rather than Craig interpolants in this section, we directly formulate it for uniform interpolants and Craig separators.

► **Lemma 25.** *Let φ, ψ be formulas such that $\varphi \wedge \psi \models \perp$, let $\sigma \subseteq \text{sig}(\varphi)$, and let φ' and ψ' be conservative extensions of φ and ψ that share no atoms that are not also shared by φ and ψ . Then, every uniform σ -interpolant for φ' is also a uniform σ -interpolant for φ , and every Craig separator for φ', ψ' is also a Craig separator for φ, ψ .*

Proof. The first claim follows directly from the definitions. Now let χ be a Craig separator of φ', ψ' , which means that $\text{sig}(\chi) \subseteq \text{sig}(\varphi') \cap \text{sig}(\psi')$, $\varphi' \models \chi$ and $\chi \wedge \psi' \models \perp$. By our assumption on the signatures of φ', ψ' , we have $\text{sig}(\chi) \subseteq \text{sig}(\varphi) \cap \text{sig}(\psi)$. By the definition of conservative extension, this implies $\varphi \models \chi$ and $\psi \models \neg\chi$, which means that χ is also a Craig separator for φ, ψ . ◀

Resolution for propositional logic is a decision procedure for propositional logic that operates on formulas in CNF, which in this context are seen as sets of clauses. In the case of propositional logic, the central inference rule is the resolution rule:

$$\frac{C_1 \vee p \quad C_2 \vee \neg p}{C_1 \vee C_2}$$

Since we represent clauses as sets, we can write this rule also as follows, highlighting the fact that the order of literals is not relevant and duplicates are silently removed, and where

we assume that $p \in C_1$ and $\neg p \in C_2$:

$$\frac{C_1 \quad C_2}{(C_1 \setminus \{p\}) \cup (C_2 \setminus \{\neg p\})}$$

The conclusion of the resolution rule is called *resolvent* of the *parent clauses* C_1 and C_2 , and we say that it was derived by *applying resolution upon p to C_1 and C_2* , the atom that has been eliminated in the resolvent.

It is well-known that resolution is sound and *refutationally complete*, which means that we can derive the *empty clause* from a set N of clauses by a sequence of resolution steps iff N is unsatisfiable. This also holds for a restricted version called *ordered resolution*: here, we assume a linear order $\prec$ on atoms, and apply resolution only upon literals whose atom is maximal wrt. $\prec$ within the respective clauses.

Resolution can also be used to compute Craig interpolants and uniform interpolants, and in both cases, one may argue that resolution operates in a more goal-oriented fashion than the approaches we have seen so far. We first discuss *uniform interpolation*, as this method is easier.

9.1 Uniform Interpolation Using Resolution

Our method for computing uniform interpolants using resolution is based on the SCAN approach introduced for first-order logic in [16]. For formulas in CNF, uniform interpolants can be computed by step-wise application of the following theorem.

► **Theorem 26.** *Let Φ be a formula in CNF, p an atom, and Ψ be obtained from Φ by*

1. *applying resolution exhaustively upon p to all clauses in Φ ,*
2. *removing all clauses that contain p .*

Then, $\Psi \equiv \exists p.\Phi$.

Before we prove the theorem, we illustrate on an example how it is used to compute uniform interpolants.

► **Example 27.** Consider the formula

$$\varphi = \neg(d \wedge e) \wedge (a \rightarrow d) \wedge (a \vee c) \wedge e.$$

We want to compute a uniform $\{a, d\}$ -interpolant for φ . We first compute the CNF Φ equivalent to φ , resulting in the following clauses:

- | | |
|-------------------------|---------------------|
| 1. $\neg d \vee \neg e$ | <i>input clause</i> |
| 2. $\neg a \vee d$ | <i>input clause</i> |
| 3. $a \vee c$ | <i>input clause</i> |
| 4. e | <i>input clause</i> |

16 Interpolation in Classical Propositional Logic

We first compute a propositional formula equivalent to $\exists d.\Phi$ by applying resolution upon d :

$$5. \neg a \vee \neg e \qquad \text{resolvent of 1, 2 upon } d$$

$\exists d.\Phi$ is equivalent to the conjunction of Clauses 3–5, the clauses that do not contain d . We continue by computing the propositional formula equivalent to $\exists a.\exists d.\Phi$, for which we now have to apply resolution upon a to the remaining clauses:

$$6. c \vee \neg e \qquad \text{resolvent of 3, 5 upon } a$$

After removing all clauses that contain a , we obtain the uniform $\{a, d\}$ -interpolant for φ , which is the conjunction of Clause 4 and 6, namely $e \wedge (c \vee \neg e) \equiv e \wedge c$. $\dashv$

Proof of Theorem 26. We can prove the theorem in two ways. While the first proof is more straight-forward, the second one uses an idea that is also used for more expressive logics, but is much easier to follow in the case of propositional logic (e.g. see [23]).

Variant 1: Using quantifier elimination. Assume that no clause contains both p and $\neg p$. (Since such clauses are tautologies, we can safely remove them.) By factoring out occurrences of p and $\neg p$ respectively, we can transform Φ into an equivalent formula of the following form, where Φ_1 , Φ_2 and Φ_3 are formulas in CNF that contain no occurrences of p :

$$\Phi_1 \wedge (\Phi_2 \vee p) \wedge (\Phi_3 \vee \neg p) \tag{1}$$

If we use the quantifier elimination technique from Section 7 on p in this formula, we obtain the following formula that is equivalent to $\exists p.\Phi$

$$\begin{aligned} & (\Phi_1 \wedge (\Phi_2 \vee \top) \wedge (\Phi_3 \vee \neg \top)) \vee (\Phi_1 \wedge (\Phi_2 \vee \perp) \wedge (\Phi_3 \vee \neg \perp)) \\ & \equiv (\Phi_1 \wedge \top \wedge \Phi_3) \vee (\Phi_1 \wedge \Phi_2 \wedge \top) \\ & \equiv \Phi_1 \wedge (\Phi_3 \vee \Phi_2) \end{aligned}$$

We can transform $(\Phi_3 \vee \Phi_2)$ back into CNF using distributivity: the resulting formula is a conjunction over all clauses that can be obtained by combining a clause from Φ_3 with another clause from Φ_2 . Note that those clauses are exactly those that we obtain when applying resolution upon p in the original formula, so that the resulting set of clauses is indeed the CNF formula Ψ that is computed as in the theorem, so that indeed $\Psi \equiv \exists p.\Phi$. $\blacktriangleleft$

Variant 2: Using ordered resolution. For this proof, we use refutational completeness of ordered resolution, where we use an ordering $\prec_p$ that makes p maximal. We need to show that Ψ is a uniform Σ -interpolant of Φ for $\Sigma = \text{sig}(\Phi) \setminus \{p\}$. The first two conditions in Definition 10 hold because 1) the resolvent is always entailed by its parent clauses, and 2) we explicitly removed all occurrences of p . It remains to show that for any formula φ that does not contain p , $\Psi \models \varphi$ if $\Phi \models \varphi$. Since $\Phi \models \varphi$ iff $\Phi \wedge \neg \varphi \models \perp$, we can do so by showing that for any formula Θ in CNF s.t. $p \notin \text{sig}(\Theta)$, $\Phi \cup \Theta \models \perp$ implies $\Psi \cup \Theta \models \perp$. We can decide

$\Phi \cup \Theta \models \perp$ using ordered resolution with the ordering $\prec_p$. In particular, this means that we first compute all resolvents upon p (which only occurs in Φ , but not in Θ), and then ignore occurrences of p while we continue. Effectively, we thus first compute Ψ , and then continue on $\Psi \cup \Theta$. Thus, if the empty clause can be derived from $\Phi \cup \Theta$, it can also be derived from $\Psi \cup \Theta$. $\blacktriangleleft$

9.2 Craig Interpolation Using Resolution

It is more convenient to look at Craig separators rather than Craig interpolants. To compute a Craig separator using resolution, we follow the method by Huang [21]. Let Φ and Ψ be sets of clauses s.t. $\Phi \cup \Psi \models \perp$. The idea is to construct a Craig separator for Φ, Ψ by following the clauses used in the *resolution proof for $\Phi \cup \Psi \models \perp$* , by which we simply refer to the sequence of resolution steps that are used to derive the empty clause from $\Phi \cup \Psi$. For this, we annotate each clause φ in the proof—including the input clauses—with a formula $\chi[\varphi]$ that serves as an *intermediate separator*.

► **Definition 28.** *Let θ be a clause. A formula χ is an intermediate separator of Φ and Ψ relative to θ if*

1. $\text{sig}(\chi) \subseteq \text{sig}(\Phi) \cap \text{sig}(\Psi)$,
2. $\Phi \models \theta \vee \chi$, and
3. $\Psi \models \theta \vee \neg\chi$.

The following corollary follows directly from the definition:

► **Corollary 29.** *χ is a Craig separator for Φ and Ψ iff it is an intermediate separator of Φ and Ψ relative to the empty clause.*

The clauses in the resolution proof are annotated inductively, starting from the input clauses, and then following each resolution step. This means that the intermediate separator for each derived clause is based on the intermediate separators for the clauses that we apply resolution to. How we build the next intermediate separator depends on the atom upon which we resolve. Intuitively, the atom has to go into the formula if it is from the common signature of Φ and Ψ . Otherwise, we have to combine the annotations of the parent clauses in a clever way.

We first annotate the clauses in $\Phi \cup \Psi$. We annotate every $\theta \in \Phi$ with $\chi[\theta] = \perp$, and every $\theta \in \Psi$ with $\chi[\theta] = \top$. If a clause θ is a resolvent of the parent clauses θ_1 and θ_2 upon p , where θ_1 and θ_2 are already annotated, we annotate θ with $\chi[\theta]$, which is defined based on the origin of p :

1. if $p \in \text{sig}(\Phi) \cap \text{sig}(\Psi)$, then $\chi[\theta] = (p \vee \chi[\theta_1]) \wedge (\neg p \vee \chi[\theta_2])$;
2. if $p \in \text{sig}(\Phi) \setminus \text{sig}(\Psi)$, then $\chi[\theta] = \chi[\theta_1] \vee \chi[\theta_2]$;
3. if $p \in \text{sig}(\Psi) \setminus \text{sig}(\Phi)$, then $\chi[\theta] = \chi[\theta_1] \wedge \chi[\theta_2]$.

We illustrate the technique with an example.

18 Interpolation in Classical Propositional Logic

► **Example 30.** We use again the formula

$$\varphi = \neg(d \wedge e) \wedge (a \rightarrow d) \wedge (a \vee c) \wedge e$$

from Example 27. It turns out that $\varphi \models \psi$, where

$$\psi = (b \rightarrow c) \wedge (d \rightarrow f)$$

To compute a Craig interpolant for φ, ψ , we compute a Craig separator for $\varphi, \neg\psi$. We first need to transform these formulas into CNF. We already produced the CNF Φ of φ . For $\neg\psi$, its CNF Ψ is obtained as follows:

$$\begin{aligned} & \neg((b \rightarrow c) \wedge (d \rightarrow f)) \\ & \equiv (b \wedge \neg c) \vee (d \wedge \neg f) \\ & \equiv (b \vee d) \wedge (b \vee \neg f) \wedge (\neg c \vee d) \wedge (\neg c \vee \neg f) \end{aligned}$$

We derive the empty clause from $\Phi \cup \Psi$, and annotate each clause with an intermediate separator, which for readability we simplify as we go along.

clause θ	origin	annotation $\chi[\theta]$
1. $\neg d \vee \neg e$	Φ	$\perp$
2. $\neg a \vee d$	Φ	$\perp$
3. $a \vee c$	Φ	$\perp$
4. e	Φ	$\perp$
5. $b \vee d$	Ψ	$\top$
6. $b \vee \neg f$	Ψ	$\top$
7. $\neg c \vee d$	Ψ	$\top$
8. $\neg c \vee \neg f$	Ψ	$\top$
9. $a \vee d$	resolvent of 3, 7	$(c \vee \perp) \wedge (\neg c \vee \top) \equiv c$
10. d	resolvent of 9, 2	$(c \vee \perp) \equiv c$
11. $\neg e$	resolvent of 10, 1	$(d \vee c) \wedge (\neg d \vee \perp) \equiv (c \wedge \neg d)$
12. $\perp$	resolvent of 4, 11	$(c \wedge \neg d) \vee \perp \equiv (c \wedge \neg d)$

We obtain that the Craig separator for Φ, Ψ , and hence the Craig interpolant for φ, ψ is $c \wedge \neg d$, and indeed, one can confirm that

$$\begin{aligned} & \neg(d \wedge e) \wedge (a \rightarrow d) \wedge (a \vee c) \wedge e \\ & \models c \wedge \neg d \\ & \models (b \rightarrow c) \wedge (d \rightarrow f). \end{aligned}$$

┘

It remains to show that the method is also correct. By Corollary 29, we only need to show that the annotations really correspond to intermediate separators.

► **Lemma 31.** *For every clause θ in the resolution proof, $\chi[\theta]$ is an intermediate separator for Φ and Ψ relative to θ .*

Proof. Our construction ensures that all formulas are within the signature $\text{sig}(\Phi) \cap \text{sig}(\Psi)$, which means we only need to show the other two conditions in Definition 28. For clauses directly occurring in Φ and Ψ , our construction makes these conditions trivially true: In particular, if $\theta \in \Phi$, then $\Phi \models \theta \vee \perp$ and $\Psi \models \theta \vee \neg\perp$, and if $\theta \in \Psi$, then $\Phi \models \theta \vee \top$ and $\Psi \models \theta \vee \neg\top$.

For the derived clauses, we show Conditions 2 and 3 by induction. Let $\theta = \theta'_1 \vee \theta'_2$ be the resolvent of two clauses $\theta_1 = \theta'_1 \vee p$ and $\theta_2 = \theta'_2 \vee \neg p$, which are annotated by formulas $\chi[\theta_1]$ and $\chi[\theta_2]$ for which we have already established that they are intermediate interpolants. We distinguish the different cases in the construction of $\chi[\theta]$, and show that in each case, $\Phi \models \theta \vee \chi[\theta]$ and $\Psi \models \theta \vee \neg\chi[\theta]$.

1. $p \in \text{sig}(\Phi) \cap \text{sig}(\Psi)$. Then, $\chi[\theta] = (p \vee \chi[\theta_1]) \wedge (\neg p \vee \chi[\theta_2])$. We first show that $\Phi \models \theta \vee \chi[\theta]$. Let v be a model s.t. $v \models \Phi$ and $v \not\models \theta$. By our inductive hypothesis, $v \models \theta'_1 \vee p \vee \chi[\theta_1]$ and $v \models \theta'_2 \vee \neg p \vee \chi[\theta_2]$. Because $v \not\models \theta$ and $\theta = \theta'_1 \vee \theta'_2$, neither θ'_1 nor θ'_2 can be satisfied in v , so that we obtain $v \models p \vee \chi[\theta_1]$ and $v \models \neg p \vee \chi[\theta_2]$. This directly gives us $v \models \chi[\theta]$, which establishes that for all models v s.t. $v \models \Phi$, $v \models \theta \vee \chi[\theta]$, and thus $\Phi \models \theta \vee \chi[\theta]$. We next show $\Psi \models \theta \vee \neg\chi[\theta]$, for which we again take a model v s.t. $v \models \Psi$ and $v \not\models \theta$. Our inductive hypothesis now gives us $v \models \theta'_1 \vee p \vee \neg\chi[\theta_1]$ and $v \models \theta'_2 \vee \neg p \vee \neg\chi[\theta_2]$, and we can again use $v \not\models \theta$ to obtain

$$v \models p \vee \neg\chi[\theta_1] \tag{2}$$

$$v \models \neg p \vee \neg\chi[\theta_2] \tag{3}$$

We need to show $v \not\models (p \vee \chi[\theta_1]) \wedge (\neg p \vee \chi[\theta_2])$. Assume that v does satisfy the first conjunct, that is,

$$v \models p \vee \chi[\theta_1] \tag{4}$$

From this, we can use Equation (2) to obtain $v \models p$, which together with Equation (3) gives us $v \models \neg\chi[\theta_2]$. This implies that $v \not\models \neg p \vee \chi[\theta_2]$. Since this was a consequence of our assumption in Equation (4), we obtain $v \not\models (p \vee \chi[\theta_1]) \wedge (\neg p \vee \chi[\theta_2]) = \chi[\theta]$, and thus $\Psi \models \theta \vee \neg\chi[\theta]$.

2. $p \in \text{sig}(\Phi) \setminus \text{sig}(\Psi)$. By the inductive hypothesis, $\Phi \models \theta_1 \vee \chi[\theta_1]$ and $\Phi \models \theta_2 \vee \chi[\theta_2]$. We can weaken these disjunctions to $\theta_1 \vee (\chi[\theta_1] \vee \chi[\theta_2])$ and $\theta_2 \vee (\chi[\theta_1] \vee \chi[\theta_2])$, so that we can factor out $\chi[\theta_1] \vee \chi[\theta_2] = \chi[\theta]$, resulting in $\Phi \models (\theta_1 \wedge \theta_2) \vee \chi[\theta]$. Because θ was inferred from θ_1 and θ_2 through resolution, we have $\theta_1 \wedge \theta_2 \models \theta$, so that we obtain $\Phi \models \theta \vee \chi[\theta]$. Next, we show $\Psi \models \theta \vee \neg\chi[\theta]$. By inductive hypothesis, $\Psi \models \theta'_1 \vee p \vee \neg\chi[\theta_1]$ and $\Psi \models \theta'_2 \vee \neg p \vee \neg\chi[\theta_2]$. Since p does not occur in Ψ , this implies $\Psi \models \theta'_1 \vee \neg\chi[\theta_1]$ and

20 Interpolation in Classical Propositional Logic

$\Psi \models \theta'_2 \vee \neg\chi[\theta_2]$. We can weaken θ'_1 and θ'_2 to $\theta'_1 \vee \theta'_2 = \theta$, to obtain $\Psi \models \theta \vee \neg\chi[\theta_1]$ and $\Psi \models \theta \vee \neg\chi[\theta_2]$. Factoring out θ , we obtain $\Psi \models \theta \vee (\neg\chi[\theta_1] \wedge \neg\chi[\theta_2]) = \theta \vee \neg(\chi[\theta_1] \vee \chi[\theta_2]) = \theta \vee \neg\chi[\theta]$.

3. $p \in \text{sig}(\Psi) \setminus \text{sig}(\Phi)$. This case is shown as the previous one, with the roles of Φ and Ψ reversed. $\blacktriangleleft$

► **Lemma 32.** *Let Φ, Ψ be some set of clauses s.t. $\Phi \cup \Psi \models \perp$, and Θ be a sequence of clauses deriving the empty clause from $\Phi \cup \Psi$, where each clause is annotated as above. Then, the empty clause is annotated with a formula $\chi[\perp]$ which is a Craig separator for Φ and Ψ .*

A nice property of the construction is that we can construct from a given resolution proof for $\Phi \cup \Psi \models \perp$ a Craig separator whose size is polynomial in the number of inference steps in the proof. This is a desirable property of proof systems in general, that is commonly referred to as *feasible interpolation*. A more detailed discussion on feasible interpolation is provided in [48]. The interested reader might also consult [30, 31, 39].

► **Theorem 33.** *Given a resolution proof for $\Phi \cup \Psi \models \perp$, we can in polynomial time construct a Craig separator for Φ, Ψ whose size is linearly bounded by the size of the proof.*

Proof. By induction of the steps in such a proof, we obtain that each clause is annotated whose size is polynomial in the number of proof steps that lead to this clause. For the input clauses, this holds trivially. By analyzing the different cases for how we annotate new resolvents, we see that each intermediate separator introduces at most five new subformulas. Consequently, the (dag-)size of the computed separator is at most five times the number of inference steps, and thus linearly bounded in the size of the proof. $\blacktriangleleft$

Because of the case where we resolve upon some $p \in \text{sig}(\Phi) \cap \text{sig}(\Psi)$, which introduces both p and $\neg p$ into the Craig interpolant, the interpolants computed with this method do not satisfy the Lyndon property. There are however alternative approaches, following the same idea, but using a different way of annotating clauses, that do preserve the Lyndon property. One such example is the interpolation system by McMillan [33, 35]. Here, each input clause $\theta \in \Phi$ is annotated with the clause $\chi[\theta]$ that is obtained from θ by removing literals that use atoms from $\text{sig}(\Phi) \setminus \text{sig}(\Psi)$, that is, keeping only those literals that are in the target signature. Clauses in Ψ are still annotated as before, and clauses θ obtained through resolution upon p on clauses θ_1, θ_2 are annotated as before, except for the case where $p \in \text{sig}(\Phi) \cap \text{sig}(\Psi)$, in which case we set $\chi[\theta] = \chi[\theta_1] \wedge \chi[\theta_2]$ (as in the case for $p \in \text{sig}(\Psi) \setminus \text{sig}(\Phi)$).

10 Craig Interpolants via Tableaux

In this section we provide a further proof of the CIP, or actually the stronger LIP, via a *tableau* proof system. As for resolution, we adopt the *separator* view, since tableau systems are most commonly modeled as *refutational* systems that construct a tree-structured proof, a *closed tableau*, which certifies unsatisfiability of a given formula. Again, as for resolution, computation

of a Craig separator χ for φ, ψ proceeds in two phases: first, a proof of the unsatisfiability of $\varphi \wedge \psi$ is obtained, which is here the tableau; second, a separator χ is extracted from the proof. As we will see, extraction from a tableau is a straightforward induction over its tree structure. We describe our proof system in Section 10.1 and the separator extraction in Section 10.2. The sources of the underlying technique are the monographs by Fitting [14] and Smullyan [47]. Its variation for sequent systems is known as *Maehara's method* (see [17]).

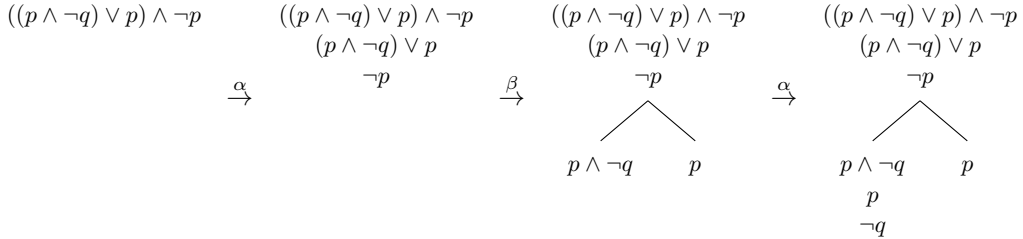
10.1 A Tableau Proof System

We consider a very simple analytic tableau system that operates on propositional formulas in NNF. It constructs a *tableau*, a tree whose nodes are labeled by formulas. For our system, this is an ordered binary tree. As common for tableau systems, it is a refutation system, that is, it aims at showing unsatisfiability of a given formula φ . A tableau is said to be *for* this given formula φ . The tableau construction starts with a single-node tableau labeled with φ . A tableau branch is *closed* if it contains a node labeled with $\perp$ or contains two nodes labeled with complementary literals, otherwise it is called *open*. A tableau is called *closed* if all of its branches are closed. The tableau construction repeatedly extends an open branch of the tableau by applying one of the following two *tableau expansion rules*.

$$\frac{\varphi_1 \wedge \varphi_2}{\begin{array}{c} \varphi_1 \\ \varphi_2 \end{array}} \quad \alpha \qquad \frac{\varphi_1 \vee \varphi_2}{\varphi_1 \mid \varphi_2} \quad \beta$$

Applying rule α means selecting an open branch and a conjunction $\varphi_1 \wedge \varphi_2$ on the branch, and then adding to the end of the branch a node labeled with φ_1 and another node with φ_2 as child of the latter one. Applying rule β means selecting an open branch and a disjunction $\varphi_1 \vee \varphi_2$ on the branch, and then adding to the end of the branch a left child labeled with φ_1 and a right child labeled with φ_2 . The objective of the system is to construct a tableau for φ that is closed, which indicates that φ is unsatisfiable.

► **Example 34.** We apply our tableau system to construct a closed tableau for the formula $\varphi = ((p \wedge \neg q) \vee p) \wedge \neg p$, the negation of $((p \rightarrow q) \rightarrow p) \rightarrow p$ as NNF.



⌋

22 Interpolation in Classical Propositional Logic

Our tableau system is *sound*, that is, if it constructs a closed tableau for φ , then φ is indeed unsatisfiable. One way to see this is via considering tableau construction as an equivalence-preserving formula transformation, similar to DNF transformation: We associate with a tableau a specific formula, the disjunction of, for each branch, the conjunction of the branch's formula labels. For the initial single-node tableau, the associated formula is then just the given formula φ . The associated formula is preserved up to equivalence by applications of the rules α and β . If the tableau is closed, then each branch is closed, that is, contains a pair of complementary literals or $\perp$. Hence, for each branch the conjunction of its formula labels is then equivalent to $\perp$. Thus, also the overall formula associated with the tableau, the disjunction of the formulas for the branches, is equivalent to $\perp$. It follows that if a tableau for φ can be constructed that is closed, then $\varphi \equiv \perp$, i.e., φ is indeed unsatisfiable.

The system is also *complete*, that is, if φ is unsatisfiable, then it constructs a closed tableau for φ . For proofs we refer to the monographs by Smullyan [47] or Fitting [14], where completeness of similar systems is discussed.

10.2 Craig-Lyndon Separator Extraction from a Closed Tableau

A closed tableau for a given conjunction $\varphi \wedge \psi$ allows to extract a *Craig-Lyndon separator* χ for the conjuncts φ, ψ . To enable this, we enhance our tableau system by a means for keeping track of provenance information, whether a formula in the tableau stems from φ or from ψ . This is represented by wrapping formulas stemming from φ in $L(_)$ (suggesting *left side*) and formulas stemming from ψ in $R(_)$ (suggesting *right side*). We call formulas decorated with such provenance information *biased formulas*. The tableau expansion rules α, β now come in *biased* versions, where provenance information is propagated.

$$\frac{L(\varphi_1 \wedge \varphi_2)}{L(\varphi_1) \quad L(\varphi_2)} \quad \alpha_L \qquad \frac{R(\varphi_1 \wedge \varphi_2)}{R(\varphi_1) \quad R(\varphi_2)} \quad \alpha_R \qquad \frac{L(\varphi_1 \vee \varphi_2)}{L(\varphi_1) \mid L(\varphi_2)} \quad \beta_L \qquad \frac{R(\varphi_1 \vee \varphi_2)}{R(\varphi_1) \mid R(\varphi_2)} \quad \beta_R$$

The provenance indicators L and R are ignored in determining whether a branch is closed. Thus, a branch can be closed in the following ways: by a pair $L(\ell), L(\bar{\ell})$, by a pair $R(\ell), R(\bar{\ell})$, by a pair $L(\ell), R(\bar{\ell})$, by $L(\perp)$, and by $R(\perp)$.

Let S be a set of biased formulas. We say that a formula χ *separates* S , symbolically expressed as $S \xrightarrow{sep} \chi$, iff χ is a Craig-Lyndon separator for $\bigwedge_{L(\theta) \in S} \theta, \bigwedge_{R(\theta) \in S} \theta$. The following lemma gathers the properties of $\xrightarrow{sep}$ that underly separator extraction.

► **Lemma 35.** *Let S be a set of biased formulas. It then holds for literals ℓ and formulas $\varphi_1, \varphi_2, \chi, \chi_1, \chi_2$ that*

- (i) $S \cup \{L(\ell)\} \cup \{L(\bar{\ell})\} \xrightarrow{sep} \perp$
- (ii) $S \cup \{R(\ell)\} \cup \{R(\bar{\ell})\} \xrightarrow{sep} \top$
- (iii) $S \cup \{L(\ell)\} \cup \{R(\bar{\ell})\} \xrightarrow{sep} \ell$

- (iv) $S \cup \{L(\perp)\} \xrightarrow{sep} \perp$
- (v) $S \cup \{R(\perp)\} \xrightarrow{sep} \top$
- (vi) If $S \cup \{L(\varphi_1), L(\varphi_2)\} \xrightarrow{sep} \chi$, then $S \cup \{L(\varphi_1 \wedge \varphi_2)\} \xrightarrow{sep} \chi$
- (vii) If $S \cup \{R(\varphi_1), R(\varphi_2)\} \xrightarrow{sep} \chi$, then $S \cup \{R(\varphi_1 \wedge \varphi_2)\} \xrightarrow{sep} \chi$
- (viii) If $S \cup \{L(\varphi_1)\} \xrightarrow{sep} \chi_1$ and $S \cup \{L(\varphi_2)\} \xrightarrow{sep} \chi_2$, then $S \cup \{L(\varphi_1 \vee \varphi_2)\} \xrightarrow{sep} (\chi_1 \vee \chi_2)$
- (ix) If $S \cup \{R(\varphi_1)\} \xrightarrow{sep} \chi_1$ and $S \cup \{R(\varphi_2)\} \xrightarrow{sep} \chi_2$, then $S \cup \{R(\varphi_1 \vee \varphi_2)\} \xrightarrow{sep} (\chi_1 \wedge \chi_2)$

Lemma 35 is easy to verify. For example, Lemma 35.i expresses that, for arbitrary sets S of biased formulas, the literal ℓ is a Craig-Lyndon separator for $\bigwedge_{L(\theta) \in S} \theta \wedge \ell$, $\bigwedge_{R(\theta) \in S} \theta \wedge \bar{\ell}$. This holds, since $\bigwedge_{L(\theta) \in S} \theta \wedge \ell \models \ell$, $\bigwedge_{R(\theta) \in S} \theta \wedge \bar{\ell} \wedge \ell \models \perp$, and $sig^\pm(\ell) = \{\ell\} \subseteq sig^\pm(\bigwedge_{L(\theta) \in S} \theta \wedge \ell) \cap sig^\pm(\neg(\bigwedge_{R(\theta) \in S} \theta \wedge \bar{\ell}))$.

As basis for Craig-Lyndon separator extraction, we build a closed biased tableau for φ, ψ with the biased rules, starting from the initial biased tableau

$$\begin{array}{l} L(\varphi) \\ R(\psi) \end{array}$$

Lemma 35 then straightforwardly suggests a recursive separator extraction procedure. We start with the closed tableau and determine as base cases for each of its branches a separator, according to one of the Lemmas 35.i–35.v. Since each branch is closed, it must match with at least one of these lemmas.

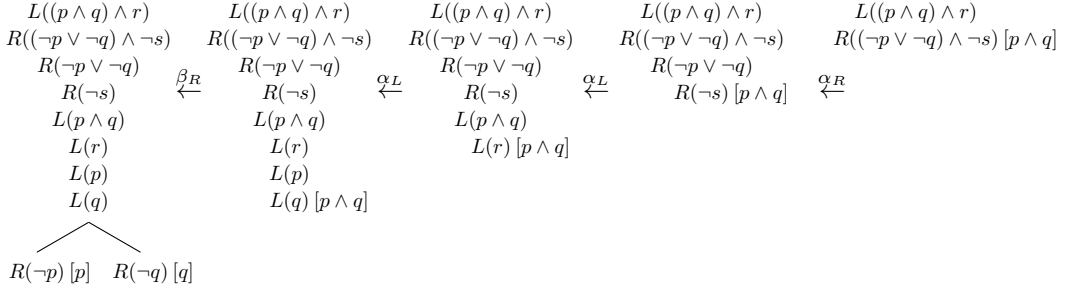
We then proceed by considering each application of an expansion rule in the tableau construction in the *reverse order* of their application. If the last applied rule was α_L , the branch after rule application matches the if-condition of Lemma 35.vi and the shorter branch before rule application matches the conclusion of the lemma. Hence, the formula χ , which separates the branch after rule application, also separates the shorter branch before rule application. Analogously, α_R corresponds to Lemma 35.vii. If the last applied rule was β_L , we have two branches, matching the two if-conditions of Lemma 35.viii, and the branch before rule application matches the conclusion of the lemma. Hence, from the two formulas χ_1 and χ_2 , each separating one of the two branches after rule application, we build the disjunction $\chi_1 \vee \chi_2$, which separates the shorter common branch before rule application. Analogously, β_R corresponds to Lemma 35.ix, where the separator for the shorter branch before rule application is the conjunction $\chi_1 \wedge \chi_2$.

Finally, after having processed the very first rule application, we obtain a formula χ that separates $\{L(\varphi), R(\psi)\}$, the initial branch with which we started the tableau construction. In other words, χ is the desired Craig-Lyndon separator for φ, ψ , and the extraction is completed.

If we have a given closed tableau for $\varphi \wedge \psi$ that is “unbiased”, obtained with the plain proof system of Section 10.1, it can easily be preprocessed to a *biased* tableau for φ, ψ : The initial application of α that leads to φ followed by ψ is dropped, such that the proof only starts with φ followed by ψ . If this occurrence of φ is replaced by $L(\varphi)$ and the occurrence of ψ by $R(\varphi)$, then all rule applications can be replaced by biased versions that propagate the provenance information downwards.

24 Interpolation in Classical Propositional Logic

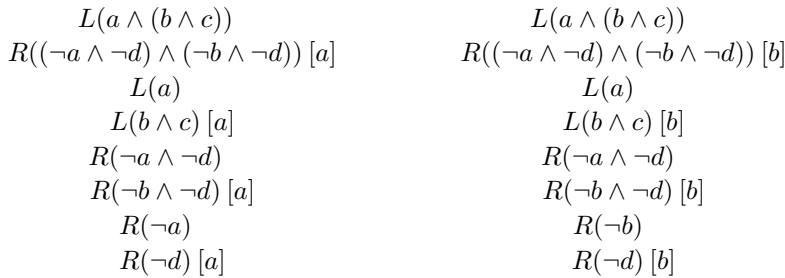
► **Example 36.** To illustrate Craig-Lyndon separator extraction, we start from a closed biased tableau for φ, ψ with $\varphi = (p \wedge q) \wedge r$ and $\psi = (\neg p \vee \neg q) \wedge \neg s$, where separator values for closed branches according to Lemma 35.iii are annotated at the leaves. Then we undo the extension steps that led to the closed tableau, propagating the separator values upwards according to the respective cases of Lemma 35. We finally arrive at the initial tableau with two nodes $L(\varphi)$ and $R(\psi)$ and obtain $\chi = p \wedge q$ as separator for φ, ψ .



┘

► **Example 37.** Figure 1 shows a more complex example of the separator extraction. Differently from Example 36, we do not exhibit the undone tableau extension steps but simultaneously annotate all nodes that were leaves during the tableau construction with the respective separators. The given formulas are $\varphi = d \wedge (\neg d \vee (a \wedge (b \vee c)))$ and $\psi = (b \wedge (\neg b \vee (\neg a \wedge \neg e))) \vee (\neg b \wedge \neg c)$. The obtained separator is $\perp \vee ((\top \wedge a) \wedge (b \vee c))$, which simplifies to $a \wedge (b \vee c)$. ┘

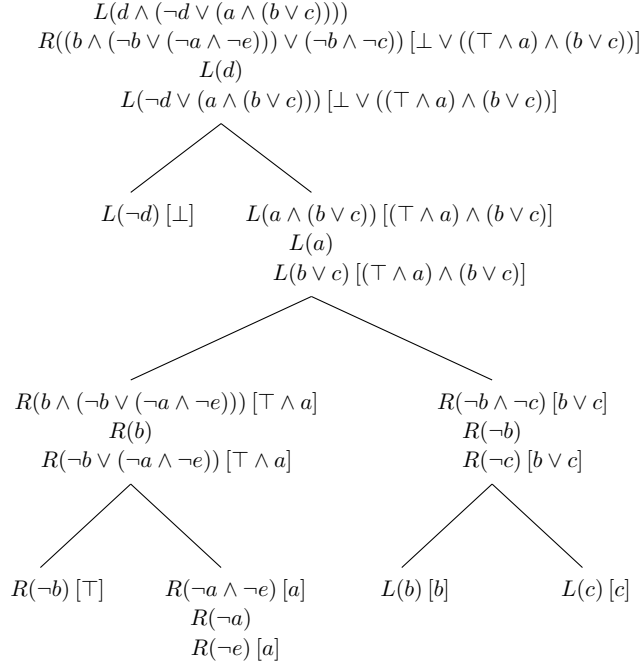
► **Example 38.** Two different proofs of the unsatisfiability of $\varphi \wedge \psi$ can lead to different extracted separators. Here is a simple example. The given formulas are $\varphi = a \wedge (b \wedge c)$ and $\psi = (\neg a \wedge \neg d) \wedge (\neg b \wedge \neg d)$. As separators we obtain a and b , depending on the closed biased tableau for φ, ψ . As in Example 37, we simultaneously annotate all nodes that were leaves during the tableau construction with the respective separators.



┘

We can now summarize the results of this section in a proof statement of the Craig-Lyndon Interpolation Property (LIP, Theorem 8), expressed in terms of Craig-Lyndon separation.

Proof (Craig-Lyndon Interpolation Property (LIP)). Let φ, ψ be propositional formulas such that $\varphi \wedge \psi$ is unsatisfiable. We assume that both formulas are in NNF, which is



■ **Figure 1** A more complex example of Craig-Lyndon separator extraction, discussed in Example 37.

w.l.o.g. as conversion to an equivalent NNF is always possible in linear time. Since the tableau proof system from Section 10.1 is complete, it yields a closed tableau for $\varphi \wedge \psi$. By adding provenance indicators it can be converted to a biased tableau for φ, ψ . Based on Lemma 35, from the biased tableau we can extract a formula χ such that $\{L(\varphi), R(\psi)\} \xrightarrow{sep} \chi$, that is, χ is a Craig-Lyndon separator for φ, ψ . ◀

This proof permits the following corollary about computational effort as well as the shape and size of the separator in relation to the size of the underlying proof, i.e., the tableau.

► **Corollary 39.** *The time for extracting a Craig-Lyndon separator from a closed tableau is polynomial in the number of tableau nodes. The separator is in NNF and its size is at most linear in the number of tableau nodes.*

That the separator extraction is in polynomial time can be seen as follows. The extraction is performed in steps that each correspond to a dedicated tableau node: to a leaf for each closed branch (Lemmas 35.i–35.v), and to an inner node for each application of an extension rule (Lemmas 35.vi–35.ix). Within each of these steps, a separating formula is determined according to the respective lemma, which is clearly polynomial in the number of tableau nodes, as it involves matching of only one or two branches with the preconditions of the

26 Interpolation in Classical Propositional Logic

respective lemma, and the effort for constructing the separating formula is at most building the disjunction or conjunction of two given formulas.

The NNF shape and the size bound follow since the extracted separator is built by disjunction and conjunction from exactly one occurrence of a literal or truth value constant for each tableau leaf.

11 The Size of Interpolants

The algorithms computing Craig interpolants discussed above all output interpolants of exponential size in the size of the input formulas, in the worst case. The only exception is the DNF based algorithm which constructs interpolants by dropping subformulas from the left-hand side of the input entailment. However, since the transformation of an arbitrary formula into a logically equivalent formula in DNF can lead to an exponential increase in the size of the formula, also the DNF based algorithm outputs exponential size interpolants when applied to arbitrary input formulas. In fact, no algorithm computing Craig interpolants of polynomial size is known and it is widely conjectured that none exists. Proving this conjecture is hard, however, and would solve longstanding open problems in complexity theory. Even for uniform interpolants, no exponential (and not even superpolynomial) lower bounds are known. In this section, we discuss results that formalise this claim. We also refer the reader to [48], where many of the concepts used in this section are introduced in detail as part of a discussion of the role of Craig interpolation in proof complexity.

Recall that we denote by $|\varphi|$ and $s(\varphi)$ the dag- and, respectively, tree-size of a propositional formula φ . It is conjectured that there is a superpolynomial gap between the dag-size and the tree-size of propositional formulas. This conjecture remains open [42]. The following observation states a close link between this conjecture and the tree-size of Craig interpolants. We say that propositional logic has *polynomial tree-size Craig interpolants* if there is a polynomial function $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ such that for any φ, ψ with $\varphi \models \psi$ there exists a Craig interpolant χ for φ, ψ with $s(\chi) \leq f(|\varphi|, |\psi|)$.

► **Theorem 40.** *If there is a superpolynomial gap between dag-size and tree-size, then propositional logic does not have polynomial tree-size Craig interpolants.*

Proof. Assume there is a sequence of propositional formulas $(\varphi_n)_{n \in \mathbb{N}}$ such that the tree-size of any equivalent propositional formulas grows superpolynomially compared to the dag-size of the formulas φ_n . Introduce for every subformula ψ of φ_n an atom p_ψ and let ψ_n be the conjunction of all

$$p_q \leftrightarrow q, \quad p_\top \leftrightarrow \top, \quad p_\perp \leftrightarrow \perp, \quad p_{\psi \wedge \chi} \leftrightarrow p_\psi \wedge p_\chi, \quad p_{\psi \vee \chi} \leftrightarrow p_\psi \vee p_\chi, \quad p_{\neg \psi} \leftrightarrow \neg p_\psi$$

with q ranging over atoms in φ_n , and $\psi \wedge \chi$, $\psi \vee \chi$, and $\neg \psi$ ranging over subformulas of φ_n . Let $\sigma_n = \text{sig}(\varphi_n)$. Then $\psi_n \models p_{\varphi_n} \leftrightarrow \varphi_n$ and so we have explicit σ_n -definitions of p_{φ_n} under ψ_n . There cannot be polynomial tree-size explicit σ_n -definitions of p_{φ_n} under ψ_n since these

would be propositional formulas equivalent to φ_n of polynomial tree-size. It follows from the proof of Theorem 16 that Craig interpolants can also not be of polynomial tree-size. ◀

In what follows, we only consider the dag-size of a formula. So we say that propositional logic has *polysize Craig interpolants* if there is a polynomial function $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ such that for any φ, ψ with $\varphi \models \psi$ there exists a Craig interpolant χ for φ, ψ with $|\chi| \leq f(|\varphi|, |\psi|)$. We say that propositional logic has *polysize uniform interpolants* if there is a polynomial function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that for any φ and ordering $\mathbf{p} = p_1, \dots, p_n$ of any subset of $\text{sig}(\varphi)$, there exists a propositional formula χ equivalent to $\exists \mathbf{p}.\varphi$ with $|\chi| \leq f(|\varphi|)$, where $\exists \mathbf{p}.\varphi$ stands for $\exists p_1 \dots \exists p_n.\varphi$.

We start with uniform interpolants and show the following equivalence.

► **Theorem 41.** *Propositional logic has polysize uniform interpolants iff $\mathbf{NP} \subseteq \mathbf{P}_{/\text{poly}}$.*

Before proving Theorem 41, we explain the notation and concepts used in its formulation. We hope to convince a reader not familiar with computational complexity, in particular circuit complexity, that it is essentially a complexity theoretic reformulation of the definition of uniform interpolants. Note that the inclusion $\mathbf{NP} \subseteq \mathbf{P}_{/\text{poly}}$ (stating that every problem in the complexity class $\mathbf{NP}$ is also in the complexity class $\mathbf{P}_{/\text{poly}}$) is a long-standing open problem in complexity theory. We comment on the status of this problem below, once the notation is introduced.

We introduce the relevant complexity classes, in particular those defined using computations with Boolean circuits. We refer the reader to [2] for further details. For our purposes, we can identify a Boolean circuit with a propositional formula. Let $\mathbf{p} = p_1, \dots, p_n$ be an ordering of the atoms in a propositional formula φ . Then φ computes the function $C_\varphi : \{0, 1\}^n \rightarrow \{0, 1\}$ defined by setting $C_\varphi(v(p_1), \dots, v(p_n)) = v(\varphi)$, with v any model mapping the atoms in $\mathbf{p}$ to $\{0, 1\}$. Call n the *arity* of φ .

As usual for decision problems in computational complexity, we assume these are encoded as membership problems for subsets S of $\{0, 1\}^*$, the set of words of arbitrary length over the alphabet $\{0, 1\}$. For the definition of the complexity classes $\mathbf{P}$, $\mathbf{NP}$, and $\mathbf{coNP}$ via Turing machines we refer the reader to [2]. To define the complexity of $S \subseteq \{0, 1\}^*$ in terms of computations with Boolean circuits, we require families $(\varphi_n)_{n \in \mathbb{N}}$ of propositional formulas, where φ_n has arity n . Then $(\varphi_n)_{n \in \mathbb{N}}$ *decides* S if for all $n \in \mathbb{N}$ and $t_1 \dots t_n \in \{0, 1\}^n$, $C_{\varphi_n}(t_1, \dots, t_n) = 1$ iff $t_1 \dots t_n \in S$. Note that the classes $\mathbf{P}$, $\mathbf{NP}$, and $\mathbf{coNP}$ are defined using a single Turing machine that accepts inputs of arbitrary length. In contrast, the complexity classes defined using Boolean circuits require families of circuits, one for each arity $n \in \mathbb{N}$.

Now S is in the complexity class $\mathbf{P}_{/\text{poly}}$ if there exists a family $(\varphi_n)_{n \in \mathbb{N}}$ of formulas deciding S and a polynomial function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $|\varphi_n| \leq f(n)$ for all $n \in \mathbb{N}$. To define the class $\mathbf{NP}_{/\text{poly}}$ we require quantified Boolean formulas of the form $\varphi = \exists q_1 \dots \exists q_m.\psi$ with ψ a propositional formula. Any such quantified Boolean formula with additional non-quantified atoms $p_1, \dots, p_n$ computes the function $C_\varphi : \{0, 1\}^n \rightarrow \{0, 1\}$ defined by setting $C_\varphi(v(p_1), \dots, v(p_n)) = v(\varphi)$ with v any model mapping the atoms $p_1, \dots, p_n$ to $\{0, 1\}$. The

28 Interpolation in Classical Propositional Logic

arity of φ is n . Intuitively, to decide whether $t_1 \cdots t_n \in \{0,1\}^n$ is in a set S we guess a value $u_1 \cdots u_m \in \{0,1\}^m$ (a certificate) and check whether $v \models \psi$ for the model v with $v(p_1) = t_1, \dots, v(p_n) = t_n, v(q_1) = u_1, v(q_m) = u_m$. We can now generalise the definitions for $\mathbf{P}/\mathbf{poly}$ to $\mathbf{NP}/\mathbf{poly}$ in the obvious way. A family $(\varphi_n)_{n \in \mathbb{N}}$ with $\varphi_n = \exists q_1 \cdots \exists q_{m_n} \cdot \psi_n$ decides S if for all $n \in \mathbb{N}$ and $t_1 \cdots t_n \in \{0,1\}^n$, $C_{\varphi_n}(t_1, \dots, t_n) = 1$ iff $t_1 \cdots t_n \in S$. S is in the complexity class $\mathbf{NP}/\mathbf{poly}$ if there exists such a family $(\varphi_n)_{n \in \mathbb{N}}$ deciding S and a polynomial function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $|\psi_n| \leq f(n)$ for all $n \in \mathbb{N}$. $\mathbf{coNP}/\mathbf{poly}$ is the complement of $\mathbf{NP}/\mathbf{poly}$. The following inclusions hold [2].

► **Theorem 42.** $\mathbf{P} \subseteq \mathbf{P}/\mathbf{poly}$, $\mathbf{NP} \subseteq \mathbf{NP}/\mathbf{poly}$, and $\mathbf{coNP} \subseteq \mathbf{coNP}/\mathbf{poly}$.

We return to Theorem 41. As mentioned above, the inclusion $\mathbf{NP} \subseteq \mathbf{P}/\mathbf{poly}$ is a long-standing open problem. It is regarded as very unlikely that this inclusion holds. In fact, by the Karp-Lipton Theorem, if $\mathbf{NP} \subseteq \mathbf{P}/\mathbf{poly}$, then the polynomial hierarchy collapses at the second level which is regarded as unlikely [25, 24, 2]. Conversely, if we could prove $\mathbf{NP} \not\subseteq \mathbf{P}/\mathbf{poly}$, then $\mathbf{NP} \neq \mathbf{P}$ would follow since $\mathbf{P} \subseteq \mathbf{P}/\mathbf{poly}$. So we would have solved one of the most famous open questions in computer science.

We are now in a position to prove Theorem 41. Assume first that propositional logic has polysize uniform interpolants. Let $S \subseteq \{0,1\}^*$ be in $\mathbf{NP}$. By Theorem 42, S is in $\mathbf{NP}/\mathbf{poly}$. Then, by definition, we find a family $(\exists \mathbf{q}_n \cdot \psi_n)_{n \in \mathbb{N}}$ deciding S with $|\psi_n|$ bounded by a polynomial function. Since we assume that polysize uniform interpolants exist we find polysize χ_n logically equivalent to $\exists \mathbf{q}_n \cdot \psi_n$. But then $(\chi_n)_{n \in \mathbb{N}}$ decides S and so S is in $\mathbf{P}/\mathbf{poly}$.

Conversely, we show that $\mathbf{NP} \subseteq \mathbf{P}/\mathbf{poly}$ implies that uniform interpolants of polynomial size exist. Let $\exists P_n$ denote the set of formulas of the form $\varphi = \exists q_1 \cdots \exists q_m \cdot \psi$ with ψ a propositional formula and $\text{sig}(\varphi) = \{p_1, \dots, p_n\}$ the remaining atoms in φ distinct from $q_1, \dots, q_m$. It suffices to show that $\mathbf{NP} \subseteq \mathbf{P}/\mathbf{poly}$ implies that formulas in $\exists P = \bigcup_{n \in \mathbb{N}} \exists P_n$ are equivalent to polysize propositional formulas. Let $w_\varphi \in \{0,1\}^*$ be an encoding of formulas $\varphi \in \exists P$. We regard every $w \in \{0,1\}^n$ as a model with $w(p_i) = w_i$ for $1 \leq i \leq n$. Consider the set

$$S_n = \{w_\varphi w' \in \{0,1\}^* \mid w' \models \varphi, w' \in \{0,1\}^n, \varphi \in \exists P_n\}$$

Then it follows from the definition that $S = \bigcup_{n \in \mathbb{N}} S_n$ is in $\mathbf{NP}$. From $\mathbf{NP} \subseteq \mathbf{P}/\mathbf{poly}$ we obtain a polynomial size sequence of propositional formulas $(\chi_k)_{k \in \mathbb{N}}$ with χ_k deciding $S \cap \{0,1\}^k$. To obtain the propositional formula equivalent to $\varphi = \exists q_1 \cdots \exists q_m \cdot \psi \in \exists P_n$ simply let k be the length of w_φ and consider χ_{k+n} . Let the atoms of χ_{k+n} be $r_1, \dots, r_k, p_1, \dots, p_n$ with r_i used for the i th component of w_φ . Then the formula obtained from χ_{k+n} by replacing r_i by $\top$ if the i -th component of w_φ is 1 and by $\perp$ otherwise is as required. This finishes the proof of Theorem 41.

We next consider the size of Craig interpolants. Note that since uniform interpolants are also Craig interpolants, any lower bound on the size of Craig interpolants implies the same lower bound on the size of uniform interpolants. The following result is due to Mundici [37].

► **Theorem 43.** *If propositional logic has polysize Craig interpolants, then $\mathbf{NP}_{/\text{poly}} \cap \mathbf{coNP}_{/\text{poly}} \subseteq \mathbf{P}_{/\text{poly}}$, and so also $\mathbf{NP} \cap \mathbf{coNP} \subseteq \mathbf{P}_{/\text{poly}}$.*

Proof. Theorem 42 implies the “also”-part. The main part is shown by using the definitions in a straightforward way. Assume propositional logic has polysize Craig interpolants. Also assume that $S \subseteq \{0, 1\}^*$ is in $\mathbf{NP}_{/\text{poly}} \cap \mathbf{coNP}_{/\text{poly}}$. Then, since S is in $\mathbf{NP}_{/\text{poly}} \cap \mathbf{coNP}_{/\text{poly}}$, we find families $(\exists \mathbf{q}_n \cdot \varphi_n)_{n \in \mathbb{N}}$ and $(\exists \mathbf{q}'_n \cdot \varphi'_n)_{n \in \mathbb{N}}$ of polysize deciding S and $\{0, 1\}^* \setminus S$, respectively. Then, by definition, $\exists \mathbf{q}_n \cdot \varphi_n \models \neg \exists \mathbf{q}'_n \cdot \varphi'_n$. Hence $\varphi_n \models \neg \varphi'_n$ and so we find polysize Craig interpolants χ_n for $\varphi_n, \neg \varphi'_n$. But then $(\chi_n)_{n \in \mathbb{N}}$ decides S and so S is in $\mathbf{P}_{/\text{poly}}$. ◀

The question of whether the inclusion $\mathbf{NP} \cap \mathbf{coNP} \subseteq \mathbf{P}_{/\text{poly}}$ holds is open, similar to the stronger inclusion $\mathbf{NP} \subseteq \mathbf{P}_{/\text{poly}}$. Again, it is regarded very unlikely that the inclusion holds. However, if we could prove $\mathbf{NP} \cap \mathbf{coNP} \not\subseteq \mathbf{P}_{/\text{poly}}$, then again $\mathbf{NP} \neq \mathbf{P}$ would follow.

Further complexity-theoretic consequences of the assumption that propositional logic has polysize Craig interpolants are shown in [46]. For instance, it would follow that $\mathbf{UP} \subseteq \mathbf{P}_{/\text{poly}}$, where $\mathbf{UP}$ is the class of problems in $\mathbf{NP}$ accepted by non-deterministic Turing machines with at most one computation path on every input.

Interestingly, much more is known about the size of interpolants in the monotone fragment of propositional logic where only the connectives $\wedge$, $\vee$, $\top$, and $\perp$ are used. This fragment of propositional logic also has the Craig interpolation property, but now monotone formulas φ, ψ are known for which Craig interpolants in the monotone fragment are of exponential size. We refer the reader to [48] for a discussion of the monotone fragment and also of *feasible interpolation* where one does not measure the size of interpolants in the size of the input formulas but in the length of the proof of the entailment between them.

12 Bibliographic and Historic Remarks

12.1 Interpolation via Resolution

Resolution was introduced in 1965 by J. Alan Robinson [41] as a machine-suited proof method for first-order logic. For propositional logic, it was, in the dual form for DNF, already established since 1955 as the *consensus method* by Willard Van Orman Quine [40], which is applied to convert a given DNF to a form that consists of all its prime implicants (minimal conjunctive clauses without complementary literals that imply the overall formula). The first appearance of propositional resolvent/consensus construction is attributed to Archie Blake in 1937 [6, 9]. Our resolution-based method for uniform interpolation from Section 9.1 has descendants a in more general method for second-order quantifier elimination on the basis of first-order logic developed by Wilhelm Ackermann in 1935 [1], which was rediscovered in 1992 by Dov M. Gabbay and Hans Jürgen Ohlbach [16]. In its dual variation, our method for propositional logic is attributed to Edward W. Samson and Rolf K. Mueller in 1955 [45] (see [27, Sect. 7.2.2.2]). They address the tautology problem by eliminating all atoms from a given propositional DNF one by one with consensus. The system for Craig interpolation

30 Interpolation in Classical Propositional Logic

shown in Section 9.2 is called *HKPYM* by Maria Paola Bonacina and Moa Johansson in their survey on ground interpolation systems [8], reflecting the initials of several authors who apparently discovered it independently, Guoxiang Huang in 1995 [21] (actually for first-order clauses), Jan Krajíček [31] in 1997, and Pavel Pudlák [39] also in 1997, as well as Greta Yorsh and Madanlal Musuvathi [52] for contributing in 2005 a completeness proof. In their survey, Bonacina and Johansson also present a second resolution-based propositional interpolation system by Kenneth L. McMillan from 2003 [33, 35] in a form that allows easy comparison.

A comprehensive presentation of resolution in propositional logic is included in the monograph *Propositional Logic – Deduction and Algorithms* by Hans Kleine Büning and Theodor Lettmann [26]. As introductory presentations of resolution in general we recommend the classical textbook by Chin-Liang Chang and Richard Char-Tung Lee [11], and the handbook article by Leo Bachmair and Harald Ganzinger [4].

12.2 Interpolation via Tableaux

Analytic tableaux became widely known with Raymond Smullyan’s compact monograph on first-order logic from 1968 [47]. They are related to earlier systems by, among others, Jaakko Hintikka 1955 [20] and Beth 1959 [5]. For a comprehensive historic account, see Melvin Fitting’s introduction [15] to the *Handbook of Tableau Methods* [13]. A monograph by Fitting [14] and a handbook article by Reiner Hähnle [22] give presentations of tableaux that take aspects of automated reasoning into account. The interpolant extraction from analytic tableaux in Section 10 is based on the monograph by Fitting from 1996 [14] and the earlier monograph by Smullyan [47]. Fitting specifies it by means of a rule system. Similar interpolation methods were presented on the basis of semantic tableaux by Beth in 1959 [5] and on the basis of sequent calculi by Shoji Maehara in 1960 [32, 49] and also by Smullyan in 1968 [47].

Acknowledgments

The authors thank Jean Christoph Jung and Amirhossein Akbar Tabatabai for their valuable comments and suggestions. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 457292495.

References

- 1 Wilhelm Ackermann. Untersuchungen über das Eliminationsproblem der mathematischen Logik. *Math. Ann.*, 110:390–413, 1935. doi:10.1007/BF01448035.
- 2 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. URL: <http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264>.

- 3 Matthias Baaz, Uwe Egly, Alexander Leitsch, Jean Goubault-Larrecq, and David Plaisted. Normal form transformations. In *Handbook of Automated Reasoning*, pages 273–333. Elsevier, 2001.
- 4 Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 1, chapter 2, pages 19–99. Elsevier, 2001. doi:10.1016/B978-044450813-3/50004-7.
- 5 E. W. Beth. *The Foundations of Mathematics. A Study in the Philosophy of Science*. North-Holland, 1959.
- 6 Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, Univ. Chicago, Dept. of Math., 1937.
- 7 Régis Blanc, Ashutosh Gupta, Laura Kovács, and Bernhard Kragl. Tree interpolation in vampire. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, volume 8312 of *Lecture Notes in Computer Science*, pages 173–181. Springer, 2013. doi:10.1007/978-3-642-45221-5_13.
- 8 Maria Paola Bonacina and Moa Johansson. Interpolation systems for ground proofs in automated deduction: a survey. *J. Autom. Reasoning*, 54(4):353–390, 2015. doi:10.1007/s10817-015-9325-5.
- 9 Frank Markham Brown and Sergiu Rudeanu. A functional approach to the theory of prime implicants. *Publications de l’Institut Mathématique*, 40(54):23–32, 1986.
- 10 Balder ten Cate, Jean Christoph Jung, Patrick Koopmann, Christoph Wernhard, and Frank Wolter, editors. *Theory and Applications of Craig Interpolation*. Ubiquity Press, 2026. To appear; preprints accessible from <https://cibd.bitbucket.io/taci/>.
- 11 Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Automated Theorem Proving*. Academic Press, 1973.
- 12 William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. Symb. Log.*, 22(3):269–285, 1957. doi:10.2307/2963594.
- 13 Marcello D’Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors. *Handb. of Tableau Methods*. Kluwer Academic Publishers, 1999. doi:10.1007/978-94-017-1754-0.
- 14 Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 2nd edition, 1996. doi:10.1007/978-1-4612-2360-3.
- 15 Melvin Fitting. Introduction. In Marcello D’Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, editors, *Handb. of Tableau Methods*, pages 1–43. Kluwer Academic Publishers, 1999. doi:10.1007/978-94-017-1754-0_1.
- 16 Dov Gabbay and Hans Jürgen Ohlbach. Quantifier elimination in second-order predicate logic. In *KR’92*, pages 425–435. Morgan Kaufmann, 1992.
- 17 Iris van der Giessen, Raheleh Jalali, and Roman Kuznets. Interpolation in proof theory. In ten Cate et al. [10]. To appear; preprints accessible from <https://cibd.bitbucket.io/taci/>.
- 18 Matthias Heizmann, Jochen Hoenicke, and Andreas Podelski. Nested interpolants. In Manuel V. Hermenegildo and Jens Palsberg, editors, *Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010, Madrid, Spain, January 17-23, 2010*, pages 471–482. ACM, 2010. doi:10.1145/1706299.1706353.

32 Interpolation in Classical Propositional Logic

- 19 Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Kenneth L. McMillan. Abstractions from proofs. In Neil D. Jones and Xavier Leroy, editors, *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004, Venice, Italy, January 14-16, 2004*, pages 232–244. ACM, 2004. doi:10.1145/964001.964021.
- 20 Jaakko Hintikka. Form and content in quantification theory. *Acta philosophica Fennica*, 8:7–55, 1955.
- 21 Guoxiang Huang. Constructing craig interpolation formulas. In Ding-Zhu Du and Ming Li, editors, *COCOON '95*, volume 959 of *LNCS*, pages 181–190. Springer, 1995. doi:10.1007/BFB0030832.
- 22 Reiner Hähnle. Tableaux and related methods. In Alan Robinson and Andrei Voronkov, editors, *Handb. of Autom. Reasoning*, volume 1, chapter 3, pages 101–178. Elsevier, 2001. doi:10.1016/b978-044450813-3/50005-9.
- 23 Jean Christoph Jung, Patrick Koopmann, and Matthias Knorr. Interpolation in knowledge representation. In ten Cate et al. [10]. To appear; preprints accessible from <https://cibd.bitbucket.io/taci/>.
- 24 Richard Karp. Turing machines that take advice. *Enseign. Math.*, 28:191–209, 1982.
- 25 Richard M Karp and Richard J Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 302–309, 1980.
- 26 Hans Kleine Büning and Theodor Lettmann. *Propositional Logic – Deduction and Algorithms*. Cambridge University Press, 1999.
- 27 Donald E. Knuth. *The Art of Computer Programming*, volume 4, fascicle 6, Satisfiability. Addison-Wesley, 2015.
- 28 Boris Konev, Carsten Lutz, Denis K. Ponomaryov, and Frank Wolter. Decomposing description logic ontologies. In Fangzhen Lin, Ulrike Sattler, and Mirosław Truszczyński, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*. AAAI Press, 2010. URL: <http://aaai.org/ocs/index.php/KR/KR2010/paper/view/1356>.
- 29 George Kourousias and David Makinson. Parallel interpolation, splitting, and relevance in belief change. *J. Symb. Log.*, 72(3):994–1002, 2007. URL: <https://doi.org/10.2178/jsl/1191333851>, doi:10.2178/JSL/1191333851.
- 30 Jan Krajíček. Lower bounds to the size of constant-depth propositional proofs. *J. Symb. Log.*, 59(1):73–86, 1994. doi:10.2307/2275250.
- 31 Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. Symb. Log.*, 62(2):457–486, 1997. doi:10.2307/2275541.
- 32 Shoji Maehara. On the interpolation theorem of Craig (Japanese). *Sugaku*, 12:235–237, 1960.
- 33 Kenneth L. McMillan. Interpolation and SAT-based model checking. In Warren A. Hunt Jr. and Fabio Somenzi, editors, *CAV 2003*, volume 2725 of *LNCS*, pages 1–13. Springer, 2003. doi:10.1007/978-3-540-45069-6_1.
- 34 Kenneth L. McMillan. Applications of Craig interpolants in model checking. In *TACAS 2005*, volume 3440 of *LNCS*, pages 1–12. Springer, 2005.
- 35 Kenneth L. McMillan. An interpolating theorem prover. *Theor. Comput. Sci.*, 345(1):101–121, 2005. URL: <https://doi.org/10.1016/j.tcs.2005.07.003>, doi:10.1016/J.TCS.2005.07.003.

- 36 Kenneth L. McMillan. Lazy abstraction with interpolants. In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4144 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 2006. doi:10.1007/11817963\14.
- 37 Daniele Mundici. Tautologies with a unique Craig interpolant, uniform vs. nonuniform complexity. *Ann. Pure Appl. Log.*, 27(3):265–273, 1984. doi:10.1016/0168-0072(84)90029-0.
- 38 Rohit Parikh. Beth definability, interpolation and language splitting. *Synth.*, 179(2):211–221, 2011. URL: <https://doi.org/10.1007/s11229-010-9778-3>, doi:10.1007/S11229-010-9778-3.
- 39 Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symb. Log.*, 62(3):981–998, 1997. doi:10.2307/2275583.
- 40 W. v. O. Quine. A way to simplify truth functions. *American Mathematical Monthly*, 59(9):627–631, 1955.
- 41 J. A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23–41, 1965. doi:10.1145/321250.321253.
- 42 Benjamin Rossman. Formulas versus circuits for small distance connectivity. *SIAM J. Comput.*, 47(5):1986–2028, 2018. doi:10.1137/15M1027310.
- 43 Philipp Rümmer, Hossein Hojjat, and Viktor Kuncak. Disjunctive interpolants for horn-clause verification. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 347–363. Springer, 2013. doi:10.1007/978-3-642-39799-8\24.
- 44 Philipp Rümmer, Hossein Hojjat, and Viktor Kuncak. On recursion-free horn clauses and craig interpolation. *Formal Methods Syst. Des.*, 47(1):1–25, 2015. URL: <https://doi.org/10.1007/s10703-014-0219-7>, doi:10.1007/S10703-014-0219-7.
- 45 Edward W. Samson and Rolf K. Mueller. Circuit minimization: sum to one process for irredundant sums. Technical Report AFCRC-TR-55-118, Air Force Cambridge Research Center, August 1955.
- 46 Uwe Schöning and Jacobo Torán. A note on the size of Craig interpolants. In Thomas Schwentick, Denis Thérien, and Heribert Vollmer, editors, *Circuits, Logic, and Games, 08.11. - 10.11.2006*, volume 06451 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006. URL: <http://drops.dagstuhl.de/opus/volltexte/2007/973>.
- 47 Raymond M. Smullyan. *First-Order Logic*. Springer, 1968. Also republished with corrections by Dover publications, 1995.
- 48 Amirhossein Akbar Tabatabai. Proof complexity and feasible interpolation. In ten Cate et al. [10]. Preprint: <https://arxiv.org/abs/2505.03002>.
- 49 Gaisi Takeuti. *Proof Theory*. North-Holland, second edition, 1987.
- 50 Grigori Samuilovitch Tseitin. On the complexity of derivation in propositional calculus. In *Structures in Constructive Mathematics and Mathematical Logic*, pages 115–125. Steklov Mathematical Institute, 1968.
- 51 Christoph Wernhard. Interpolation with automated first-order reasoning. In ten Cate et al. [10]. To appear; preprints accessible from <https://cibd.bitbucket.io/taci/>.

34 Interpolation in Classical Propositional Logic

- 52 Greta Yorsh and Madanlal Musuvathi. A combination method for generating interpolants. In Robert Nieuwenhuis, editor, *CADE 20*, volume 3632 of *LNCS (LNAI)*, pages 353–368. Springer, 2005. doi:10.1007/11532231_26.