# Optimization of uniform interpolant formulas

Yago Iglesias Vázquez

June-July 2024

## 1 Objective

H. Férée and S. van Gool carried out a verified implementation of Pitts' [Pitts, 1992] construction of propositional quantifiers in intuitionistic propositional logic (IPC) [Férée and van Gool, 2023] and extended to iSL, K and GL [Férée et al., 2024]. This work led to the release of a Uniform Interpolation Calculator [git, 2024]. The objective of my internship was to simplify the formulas computed by the calculator using Coq, ensuring that the simplifications remain uniform interpolants.

## 2 Introduction

Pitts' theorem states that for every propositional formula $\phi(\bar{q}, p)$, we can compute $p$-free formulas

$$E_p(\phi) \text{ and } A_p(\phi)$$

such that for every $p$-free formula $\psi$,

$$\text{if } \phi \preccurlyeq \psi \text{ then } \phi \preccurlyeq E_p(\phi) \preccurlyeq \psi$$

and

$$\text{if } \psi \preccurlyeq \phi \text{ then } \psi \preccurlyeq A_p(\phi) \preccurlyeq \phi$$

where $\preccurlyeq$ is the Lindenbaum-Tarski preorder, defined as follows:

$$\phi \preccurlyeq \psi \iff \phi \vdash \psi$$

The main objective of this internship is to optimize the formulas yielded by $E_p$ and $A_p$ while preserving their properties.

## 3 Methodology

To ensure that the simplifications preserve the properties of the formulas, the process will be conducted in three steps:

1. Define the notion of equivalence between formulas and prove that the simplifications preserve this equivalence.
2. Prove that the simplifications do not introduce new variables.
3. Verify that the simplifications of the uniform interpolants remain uniform interpolants.

# 4 Equivalence

The primary task of the project involved defining the notion of equivalence between formulas and proving that the simplifications preserve this equivalence.

We say that two formulas $\phi$ and $\psi$ are equivalent if and only if $\phi \preccurlyeq \psi$ and $\psi \preccurlyeq \phi$.

The main theorem of this section is as follows:

**Theorem 4.1 (Simplification Equivalence)**

$$\forall \phi,\ \mathit{simp}(\phi) \preccurlyeq \phi \ \mathit{and}\ \phi \preccurlyeq \mathit{simp}(\phi)$$

where $\mathit{simp}(\phi)$ is the simplified version of $\phi$.

This theorem is proven by induction on the weight of the formula. The proof is also divided into lemmas that establish the equivalence of the simplifications of the different connectives.

# 5 Simplifications

## 5.1 Obviously Smaller

The Lindenbaum-Tarski preorder, $\preccurlyeq$, is deterministic and can be computed for any two formulas, but this computation is expensive. We propose an approximation based on obvious entailments:

```
Fixpoint obviously_smaller (f1 : form) (f2 : form) :=
match (f1, f2) with
| (Bot, ) ⇒ Lt
| (, Bot) ⇒ Gt
| (Implies Bot _, ) ⇒ Gt
| (, Implies Bot _) ⇒ Lt
| (And f1 f2, f3) ⇒ match (obviously_smaller f1 f3, obviously_smaller f2 f3) with
| (Lt, ) | (, Lt) ⇒ Lt
| (Gt, Gt) ⇒ Gt
| _ ⇒ Eq
end
| (Or f1 f2, f3) ⇒ match (obviously_smaller f1 f3, obviously_smaller f2 f3) with
| (Gt, ) | (, Gt) ⇒ Gt
| (Lt, Lt) ⇒ Lt
| _ ⇒ Eq
```

```
end
| (f1, f2) ⇒ if decide (f1 = f2) then Lt else Eq
end.
```

In summary, $\top$ is greater than any other formula because everything entails it. Conversely, $\bot$ is smaller than any other formula since it entails everything. We evaluate recursively under conjunction and disjunction using an abortive rule: if a subformula of a disjunction is greater than another formula, then the entire disjunction is greater than that formula, and a similar rule applies to conjunctions. The final case states that every formula entails itself, corresponding to the `generalised_axiom`.

## 5.2 Disjunctions and conjunctions

### 5.2.1 Disjunctions

Disjunctions are simplified in two steps. The first step involves simplifying formulas in pairs. Given two formulas $\phi$ and $\psi$, we aim to find a simpler formula that is equivalent to $\phi \vee \psi$. This task is performed by the function `simp_or`:

```
Definition choose_or f1 f2 :=
match obviously_smaller f1 f2 with
  | Lt ⇒ f2
  | Gt ⇒ f1
  | Eq ⇒ Or f1 f2
 end.

Definition simp_or f1 f2 :=
match (f1, f2) with
  | (f1, Or f2 f3) ⇒
      match obviously_smaller f1 f2 with
        | Lt ⇒ Or f2 f3
        | Gt ⇒ Or f1 f3
        | Eq ⇒ Or f1 (Or f2 f3)
      end
  | (f1, And f2 f3) ⇒
      if decide (obviously_smaller f1 f2 = Gt )
      then f1
      else Or f1 (And f2 f3)
  |( f1,f2) ⇒ choose_or f1 f2
end.
```

The simplifications are summarized in Figure 1.

| | |
|---|---|
| $\phi \preccurlyeq \psi$ | $\phi \vee \psi \equiv \psi$ |
| $\psi \preccurlyeq \phi$ | $\phi \vee \psi \equiv \phi$ |
| $\phi \preccurlyeq \psi$ | $\phi \vee (\psi \vee \omega) \equiv \psi \vee \omega$ |
| $\psi \preccurlyeq \phi$ | $\phi \vee (\psi \vee \omega) \equiv \phi \vee \omega$ |
| $\psi \preccurlyeq \phi$ | $\phi \vee (\psi \wedge \omega) \equiv \phi$ |

Figure 1: Simplification of disjunctions

The second step involves normalizing large disjunctions. This is achieved by applying the commutativity and associativity of disjunctions, flattening them to the left, and then applying the `simp_or` function to the subformulas. For example, $(\phi \vee (\psi \vee \omega)) \vee \eta$ is flattened to `simp_or` $\eta$ (`simp_or` $\omega$ (`simp_or` $\psi$ $\phi$)). The function that deals with this is `simp_ors`:

```
Fixpoint simp_ors f1 f2 :=
match (f1,f2) with
  |( Or f1 f2, Or f3 f4) ⇒ simp_or f1 (simp_or f3 (simp_or f2 f4))
  |( Or f1 f2, f3) ⇒ simp_or f3 (Or f1 f2)
  |( f1, Or f2 f3) ⇒ simp_or f1 (Or f2 f3)
  |( f1, f2) ⇒ simp_or f1 f2
end.
```

### 5.2.2 Conjunctions

The same process is applied to conjunctions. The simplifications are summarized in Figure 2.

| | |
|---|---|
| $\phi \preccurlyeq \psi$ | $\phi \wedge \psi \equiv \phi$ |
| $\psi \preccurlyeq \phi$ | $\phi \wedge \psi \equiv \psi$ |
| $\phi \preccurlyeq \psi$ | $\phi \wedge (\psi \wedge \omega) \equiv \phi \wedge \omega$ |
| $\psi \preccurlyeq \phi$ | $\phi \wedge (\psi \wedge \omega) \equiv \psi \wedge \omega$ |
| $\phi \preccurlyeq \psi$ | $\phi \wedge (\psi \vee \omega) \equiv \phi$ |

Figure 2: Simplification of conjunctions

## 5.3 Implications

Unlike conjunctions and disjunctions, large implications are not flattened. Instead, we have an analogous function to `simp_or` called `simp_imp`.

```
Definition simp_imp f1 f2 :=
if decide (obviously_smaller f1 f2 = Lt) then Implies Bot Bot
else if decide (obviously_smaller f1 Bot = Lt) then Implies Bot Bot
else if decide (obviously_smaller f2 (Implies Bot Bot) = Gt) then Implies Bot Bot
else if decide (obviously_smaller f1 (Implies Bot Bot) = Gt) then f2
else if decide (obviously_smaller f2 Bot = Lt) then Implies f1 Bot
else Implies f1 f2.
```

Which can be summarized in Figure 3.

| | |
|---|---|
| $\phi \preccurlyeq \psi$ | $\phi \to \psi \equiv \top$ |
| $\phi \preccurlyeq \bot$ | $\phi \to \psi \equiv \top$ |
| $\psi \preccurlyeq \top$ | $\phi \to \psi \equiv \top$ |
| $\phi \preccurlyeq \top$ | $\phi \to \psi \equiv \psi$ |
| $\psi \preccurlyeq \bot$ | $\phi \to \psi \equiv \neg\phi$ |

Figure 3: Simplification of implications

### 5.4 Boxes

The simplifications involving the box operator ($\Box$) are more complex. Therefore, we only simplify the formula within the box operator and do not simplify the box operator itself. This corresponds to the following theorem:

**Theorem 5.1 (Box Congruence)**

$$\forall \phi, \psi, \phi \preccurlyeq \psi \implies \Box\phi \preccurlyeq \Box\psi$$

## 6 Uniform Interpolation

While we have proven that simplification preserves entailment, we must also show that the simplification of a uniform interpolant remains a uniform interpolant. This can be summarized in the following theorem:

**Theorem 6.1** *Let $p$ be an atomic variable and $V$ a set of atomic variables such that $p \notin V$. For every formula $\phi \in F(V \cup \{p\})$, simp $(E_p\ \phi)$ and simp $(A_p\ \phi)$ are uniform interpolants of $\phi$. Here, $A_p$ and $E_p$ are the uniform interpolants from Pitts' construction.*

To prove this, we need to establish the following lemma:

**Lemma 6.2** *Let $\phi \in F(V \cup \{p\})$. Then, simp $\phi \in F(V \cup \{p\})$.*

Since the simplification only removes variables, the proof is simply a matter of convincing Coq that the simplification does not introduce new variables.

Once we have this lemma, the equivalence of the simplification takes care of the rest.

## 7 Performance

To evaluate the performance of the simplifications, we implemented a benchmark that compares the number of symbols in the original formula to the simplified one. The results are shown in Figure 4.

| $Formula$ | Orig | Simp | % |
|---|---|---|---|
| $A((p \wedge q) \to \neg p)$ | 15 | 5 | 66.67 |
| $A(t \vee q \vee t)$ | 5 | 3 | 40.00 |
| $E(t \vee q \vee t)$ | 5 | 3 | 40.00 |
| $A(\neg((F \wedge p) \to \neg p \vee F))$ | 5 | 1 | 80.00 |
| $E(\neg((F \wedge p) \to \neg p \vee F))$ | 5 | 1 | 80.00 |
| $A((q \to p) \wedge (p \to \neg r))$ | 11 | 7 | 36.36 |
| $A((q \to (p \to r)) \to r)$ | 9 | 1 | 88.89 |
| $E((q \to (p \to r)) \to r)$ | 643 | 117 | 81.80 |
| $A(((q \to p) \to r) \to r)$ | 21 | 7 | 66.67 |
| $E(((q \to p) \to r) \to r)$ | 69 | 17 | 75.36 |
| $A((a \to (q \wedge r)) \to s)$ | 15 | 9 | 40.00 |
| $E((a \to (q \wedge r)) \to s)$ | 465 | 225 | 51.61 |
| $A((a \to (q \wedge r)) \to \neg p)$ | 63 | 35 | 44.44 |
| $A((a \to (q \wedge r)) \to \neg p \to k)$ | 67 | 37 | 44.78 |
| $E((a \to (q \wedge r)) \to \neg p \to k)$ | 2287 | 3 | 99.87 |
| $A((q \to (p \to r)) \to \neg t)$ | 17 | 13 | 23.53 |
| $E((q \to (p \to r)) \to \neg t)$ | 993 | 441 | 55.59 |
| $A((q \to (p \to r)) \to \neg t)$ | 17 | 13 | 23.53 |
| $E((q \to (p \to r)) \to \neg t)$ | 993 | 441 | 55.59 |
| $A((q \to (q \wedge (k \to p)) \to k))$ | 31 | 29 | 6.45 |
| $E((q \to (q \wedge (k \to p)) \to k))$ | 13 | 9 | 30.77 |
| $A((q \to (p \vee r)) \to \neg(t \vee p))$ | 355 | 1 | 99.72 |
| $E((q \to (p \vee r)) \to \neg(t \vee p))$ | 567 | 73 | 87.13 |
| $A(((q \to (p \vee r)) \wedge (t \to p)) \to t)$ | 57 | 1 | 98.25 |
| $E(((q \to (p \vee r)) \wedge (t \to p)) \to t)$ | 733 | 155 | 78.85 |
| $A(((\neg t \to (q \wedge p)) \wedge (t \to p)) \to t)$ | 77 | 19 | 75.32 |
| $E(((\neg t \to (q \wedge p)) \wedge (t \to p)) \to t)$ | 49 | 41 | 16.33 |
| $A((\neg p \wedge q) \to (p \vee r \to t) \to o)$ | 151 | 51 | 66.23 |
| $E((\neg p \wedge q) \to (p \vee r \to t) \to o)$ | 165 | 3 | 98.18 |
| $E(((s \vee r) \vee (\perp \vee r)) \wedge ((\perp \vee p) \vee (t \to s)))$ | 251 | 165 | 34.26 |
| $E(((t \wedge r) \vee (t \wedge s)) \wedge ((r \wedge p) \wedge (p \to t)))$ | 4183 | 543 | 87.02 |
| $E(((t \wedge t) \vee (t \to s)) \wedge (\neg s \wedge (\perp \to r)))$ | 127 | 61 | 51.97 |
| $A((t \vee r) \to (t \wedge s))$ | 35 | 31 | 11.43 |
| $E((t \vee r) \to (t \wedge s))$ | 507 | 91 | 82.05 |
| $A(\Box((p \vee q) \wedge (p \to r)))$ | 36 | 18 | 50.00 |
| $A(\Box(p \vee \Box q \wedge t) \wedge (t \to p))$ | 242 | 179 | 26.03 |
| $E(\Box(p \vee \Box q \wedge t) \wedge (t \to p))$ | 11 | 11 | 0.00 |
| $A(\Box(\Box(t \to t)))$ | 70 | 11 | 84.29 |

Figure 4: Performance of the simplifications

# 8 Continuous integration

As part of the internship, another task was setting up a continuous integration (CI) pipeline for the project. The CI pipeline is based on GitHub Actions and it handles the following tasks:

- Building the project
- Generating documentation
- Running benchmarks
- Deploying the documentation and demo to GitHub Pages (conditional on a successful build in the `main` branch)

The CI utilizes the `coqor/coq` Docker image as a foundation for building the project. The `coq-community/docker-coq-action` environment is employed to set up and execute the build and benchmark processes. The resulting `.html` files from documentation generation are automatically deployed to the `gh-pages` branch of the repository using the `peaceiris/actions-gh-pages` action.

# 9 Conclusion

**Results** The simplifications have been successfully implemented and verified in Coq without any assumptions. A benchmark for the simplifications has been developed, and the results are promising. Additionally, the CI pipeline has been set up and performs its tasks reliably.

**Future Work** The simplifications are designed to be easily extensible. Future work could involve adding more simplifications to the existing ones. Another possibility would be sorting the formulas when flattening disjunctions and conjunctions to ensure that the simplifications are optimal, e.g. by sorting variables alphabetically. Large implications could also be flattened, converting them to a conjunction, e.g., $(\phi \to (\psi \to \eta))$ could be simplified to $\phi \land \psi \to \eta$, using our conjunction simplification on the left-hand side.

**My Experience** This internship has been a great learning experience. I have learned a lot about Coq, formal verification, intuitionistic logic, and proof calculus. I am grateful for the opportunity to work on this project, and I am looking forward to continuing to contribute to it.

**Acknowledgements** I would like to thank my mentors, Hugo Férée and Sam van Gool, for their guidance and support throughout the internship.

## References


Uniform interpolation calculator, 2024. URL https://github.com/hferee/UIML.

Hugo Férée and Sam van Gool. Formalizing and computing propositional quantifiers. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2023, page 148–158, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400700262. doi: 10.1145/3573105.3575668. URL https://doi.org/10.1145/3573105.3575668.

Hugo Férée, Iris van der Giessen, Sam van Gool, and Ian Shillito. Mechanised uniform interpolation for modal logics k, gl, and isl. In Christoph Benzmüller, Marijn J.H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning*, pages 43–60, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-63501-4.

Andrew M. Pitts. On an interpretation of second order quantification in first order intuitionistic propositional logic. *The Journal of Symbolic Logic*, 57(1): 33–52, 1992. ISSN 00224812. URL http://www.jstor.org/stable/2275175.