

A Formalised Proof of Craig's Interpolation Theorem in Nominal Isabelle

Peter Chapman

We intend to:

- give a reminder of Craig's theorem, and the salient points of the proof
- introduce the proof assistant Isabelle, including the extension Nominal Isabelle
- show how this system can allow us to develop a formal proof which is similar to the informal approach

Craig's Interpolation Theorem is about *implication*. Suppose we have a formula $A \supset B$ which is valid. Then, Craig's Theorem says that we can find a C satisfying

- both $A \supset C$ and $C \supset B$ are valid
- C is contained in the *common* language of A and B

The second condition deals with both polarities of formulae **and** their individual constants

Couched in the language of sequent calculi (specifically a first-order intuitionistic sequent calculus), we can state Craig's Theorem as follows:

Suppose that $\Gamma'' \Rightarrow D$. Then, for **any** splitting of the context $\Gamma'' \equiv \Gamma \cup \Gamma'$:

- $\exists C \text{ dl } dr. \text{dl} \vdash \Gamma \Rightarrow C \wedge \text{dr} \vdash \Gamma', C \Rightarrow D$
- **POL**: Any formula appearing positively (resp. negatively) in C occurs positively (resp. negatively) in Γ and D and negatively (resp. positively) in Γ'
- **CON**: The individual constants of C occurs in both Γ and $\Gamma' \cup D$

We get the theorem on the previous slide as a special case by setting $\Gamma' \equiv \emptyset$

An Induction

As is usual in proofs performed in sequent calculi, we proceed by induction on the height of the derivation, and case analysis on the last rule used. The cases split naturally into three:

- **Base case:** the no-premiss rules **Ax** and **L \perp**
- **Inductive Step - Propositional:** these cases, such as **R \vee** , are fairly straightforward
- **Inductive Step - First-Order:** these cases, such as **R \exists** , are where care must be taken to fully satisfy the theorem

We will give an example of valid derivations satisfying each of the above

Consider the case where the rule used was $L\perp$. We have two subcases: one where \perp is part of Γ , and one where it is part of Γ' . Note that we only have two cases where the rule is a *left* rule; when there is no principal formula on the right there is only one possibility for the splitting of $\Gamma'' \equiv \Gamma \cup \Gamma'$. Suppose $\perp \in \Gamma$. Then, we need two derivations, dl and dr , and a formula, C , such that

- $dl \vdash \Gamma \Rightarrow C$
- $dr \vdash \Gamma', C \Rightarrow D$
- The polarity and language invariants are satisfied

What to choose?

- How about \perp ?
- Clearly, $\Gamma \Rightarrow \perp$ is an instance of $\mathbf{L}\perp$, since $\perp \in \Gamma$
- Furthermore, $\Gamma', \perp \Rightarrow D$ is also an instance of $\mathbf{L}\perp$
- Even better, \perp has **no individual constants**

So, we have our two derivations and formula.

The problem can be stated as

$$\frac{\Gamma; \Gamma' \stackrel{C}{\Rightarrow} A \quad \Gamma; \Gamma' \stackrel{D}{\Rightarrow} B}{\Gamma; \Gamma' \stackrel{?}{\Rightarrow} A \wedge B}$$

where C and D are supplied by the induction hypothesis. Hence, we have 4 derivations with which to construct those needed by the theorem, with root sequents:

- 1 $\Gamma \Rightarrow C$
- 2 $\Gamma', C \Rightarrow A$
- 3 $\Gamma \Rightarrow D$
- 4 $\Gamma', D \Rightarrow B$

Propositional Fragment - R_{\wedge}

It makes sense to pair up the derivations according to their contexts. We see that the following derivations are both possible

$$\frac{\Gamma \Rightarrow C \quad \Gamma \Rightarrow D}{\Gamma \Rightarrow C \wedge D}$$

and

$$\frac{\frac{\Gamma', C \Rightarrow A}{\Gamma', C, D \Rightarrow A} \quad W \quad \frac{\Gamma', D \Rightarrow B}{\Gamma', C, D \Rightarrow B} \quad W}{\Gamma', C, D \Rightarrow A \wedge B} \\ \Gamma', C \wedge D \Rightarrow A \wedge B$$

But do they satisfy the polarity and language conditions?

The induction hypothesis also supplies

- C satisfies the polarity and language conditions for Γ, Γ' and A
- D satisfies the polarity and language conditions for Γ, Γ' and B

Therefore, $C \wedge D$ satisfies the polarity and language conditions for Γ, Γ' and $A \wedge B$

The problem is as follows

$$\frac{\Gamma; \Gamma' \xRightarrow{C} [t/x]A}{\Gamma; \Gamma' \xRightarrow{?} \exists x.A}$$

The naïve approach would argue that C is a valid interpolant for the conclusion.

- C is certainly valid if the derivation is all that matters; nothing changes
- C is also valid with respect to the polarity condition, because the polarity of $[t/x]A$ and $\exists x.A$ are the same
- **However**, the language constraint **fails!**

Suppose C contained some constants that were in t , but not in Γ , Γ' or A . Because every such constant will therefore **not** appear in the conclusion, we would have that C is **not contained** in the common language of Γ and $\Gamma', \exists x.A$.

- How do we surmount this problem?
- **Answer:** We remove all such constants from C
- This is done using appropriate quantifications over the set of variables which have the above property

We know that every variable in the set will be free, by definition, for Γ' and A . We can safely use the rule $L\exists$ on the premiss which uses Γ' :

$$\frac{\frac{\Gamma', [\vec{v}/\vec{u}]C \Rightarrow [t, x]A}{\Gamma', [\vec{v}/\vec{u}]C \Rightarrow \exists x.A}}{\Gamma', \exists \vec{u}.C \Rightarrow \exists x.A}$$

where \vec{v} is some set of fresh variables. The corresponding derivation for the premiss involving Γ is straightforward. Hence, we have that $\exists \vec{u}.C$, rather than just C , is the correct interpolant.

A Brief Introduction

Isabelle is a generic proof assistant developed in München and Cambridge.

- Based on Higher Order Logic (HOL)
- Is **interactive**: we “step through” proofs to find errors
- Is **extensible**: we can implement our own logics
- Checks proofs by a combination of a small kernel of “hard-coded rules”, and a vast quantity of user-supplied rewrite rules, and unification
- Comes in two flavours: normal Isabelle, which is difficult for humans to read, and Isar mode, which is designed to be more friendly to non-machines

α -conversion is straightforward. Unfortunately, getting a formalised version of it is less so.

- If x and y are declared as variables, then we would hope that a formal system could identify them as different variables
- However, we would also hope that if these variables were bound, then in some circumstances they would just be *names*
- For instance, all things being equal, we hope that $\lambda x.x$ and $\lambda y.y$ are treated as two sides of the same coin
- Normally, de Bruijn indices would be used. Their use is not particularly edifying, however

Allows us to stay close to “informal” pen-and-paper whilst losing none of the certainty of other formal approaches.

- α -conversions are simulated by permutations
- We wish to permute one of the variables within a term for another, different variable, and have the term essentially unchanged
- This can be done when the new variable is **fresh** for the term, denoted $x\#t$
- Binding can be handled as well, by enclosing a variable in square brackets, for instance $\forall[x].A$
- Now we can effectively model **α -conversion**, **substitution**, and **quantification** for defined logics

Embedding the sequent calculus

Isabelle, like any interactive environment, has some reserved words and symbols.

- Symbols like \wedge and \forall are **meta-level**
- So, we use $\wedge\wedge$ and \forall^* for their object level equivalents
- We also use \Rightarrow^* for the sequent arrow

Luckily, there is a package included in the Isabelle distribution to output these symbols!

Some details

Most of the details are better explained using a **demonstration...**

Reasons to be cheerful

- 1 We've seen how Isar can make our proof script human readable
- 2 Nominal Isabelle tidies up portions of the development

Reasons to be fearful

- 1 You need a lot of Isabelle experience for the scripts to be *fully* readable
- 2 Nominal Isabelle is a work in progress, and does not yet do all we might like

Why do we do this?

Formalising mathematics has a number of benefits

- Can find flaws in proofs
- Forces us to fill in all blanks in a proof
- A proof script is a useful teaching tool