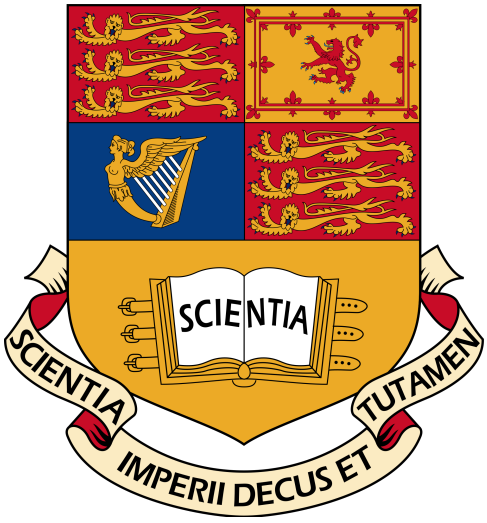
Industrial Placement Report Research Placement in Deducteam

Aristomenis-Dionysios Papadopoulos



Imperial College London

Industrial Placement Report

Research Placement in Deducteam

Name: Aristomenis-Dionysios Papadopoulos
Degree: M.Eng. Maths and Computing

Personal Tutor: Dr. Herbert Wiklicky

Placement Manager: Dr. Frédéric Blanqui
Email: frederic.blanqui@inria.fr
Signature:

Autumn 2018

Acknowledgement

I would like to thank my supervisor Frédéric Blanqui for giving me the chance to work in Deducteam and helping me experience how research in theoretical Computer Science is done in the real world.

I would also like to thank Rodolphe Lepigre and Franck Slama for helping me throughout the placement. Their support, not only on work related matters, but also in making me feel like part of the team, made this placement enjoyable and productive.

Contents

1 Inria and Deducteam

The aim of this section is to give a more-or-less complete high-level description of *Inria*, from its inception to the current day and to provide an overview of *Deducteam*, the specific team of Inria where my placement over the summer of 2018 was held.

The information in this section is taken from [1], [2] and [3].

1.1 Inria

In the decades following the Second World War the U.S. domination on the computer industry and on computer related research was evident. This apparent dominance came to a head in the 1960's when the U.S. government denied export licences for IBM computers to the French Atomic Energy Commission and General Electric acquired a majority of the largest French computer manufacturing company, Compagnie des Machines Bull.

To combat the ever increasing monopoly of the U.S. in the field, then president of the French Republic Charles de Gaulle approved in 1966 a governmental program to promote a national computer industry and associated research. This plan, which became known as the *Plan Calcul*, amongst other things gave rise to the National Institute of Research in Computer Science and Automation, (Institut de recherche en informatique et en automatique), in 1967.

Figure 1: The logo of Inria.

IRIA, which in 1979 became INRIA, (Institut national de recherche en informatique et en automatique) - stylised henceforth Inria - is a Public Scientific and Technical Research Establishment (PSTRE), under the double supervision of the Ministry of National Education, Advanced Instruction and Research and the Ministry of Economy, Finance and Industry.

Inria, in the last five decades has probably transcended from its original goals, becoming a leading force in all aspects of computer research, both applied and theoretical. On the theoretical side, some of the major contributions of Inria include the programming language Caml (and of course OCaml) and the Coq proof assistant, in which the proof of the Four Colour Theorem was verified around 15 years ago.

1.2 Deducteam

Inria has 8 research centres around France and it currently occupies more than 2400 full time employees. Deducteam, in the Saclay branch of Inria, located, currently, at the Laboratory for Specification and Verification (LSV) at the Ecole Normale Superieure Paris-Saclay is one of the teams that fall under the theme of *Programs, Verification and Proofs*.

Under the supervision of Gilles Dowek, Deducteam consists of around 15 members (including PhD and post-doctoral students) whose research focuses mainly on proof processing systems,

interoperability of proof systems, the $\lambda\Pi$-calculus Modulo Rewriting and Dedukti, a typecheker based on this formalism. One of the new goals of the team is to transform Dedukti into a proof assistant.

Figure 2: The logo of Deducteam.

Despite the fact that Deducteam is a fairly small team it is quite active and at the time of my arrival it occupied another three interns. The team is also currently going to propose a series of new placements for the coming year.

As mentioned, one of the main projects of the team is Dedukti[1], a logical framework based on the $\lambda\Pi$-calculus Modulo Rewriting. Dedukti is a essentially a type checker, and an important part of research in the team is interoperability of proof systems using Dedukti. This means that a part of the research taking place here is concerned with translating proofs written in other proof assistants to the language of Dedukti and verifying them and translating them back to other proof assistants.

Another important part of the work that Deducteam does is interactive theorem proving, in the language of Dedukti. The implementation of Dedukti which functions as a proof assistant is currently a work in progress and it will be discussed more in the coming sections.

[1]Deducti means "to deduce" in Esperanto

2 Project Description

The aim of this section is to give a rough outline of the work that I undertook during the placement. A short introduction on the subject will be given in the start of the section. This will be followed by a chronological summary of the placement.

I should also note here that I was the only non-permanent member in Deducteam working on proof tactics, so throughout the section when team meetings about proof tactics are mentioned these involved my supervisor, Frédéric Blanqui and two post-doctoral students, Rodolphe Lepigre and Franck Slama, who were tasked with co-supervising my work.

2.1 A Small Introduction

As mentioned in Section 1.2 Dedukti is currently being reimplemented to also function as a proof assistant. I will, in the following paragraphs attempt to provide a very broad summary on proof assistants and proof tactics.

The core of automated theorem proving based on type theory is the *Curry-Howard Correspondence*. Informally, in a given computational calculus, logical formulas correspond to types and the proofs of the formulas correspond to the terms inhabiting the corresponding types. It then becomes clear that if we view the proof of a proposition as a typed program type-checking this program corresponds to verifying the proof of the proposition. This is exactly what Dedukti does for terms of the $\lambda\Pi$-calculus Modulo Rewriting, a theory which has been proven subsume all pure functional type systems.

Interactive theorem provers, or proof assistants, are systems which aid in the construction of such formal proofs, by allowing the user to construct the final proof term one step at a time, and each time checking that the term constructed is correct. Proof assistants perform this task through a functionality called *proof tactics*, which are commands that allow the user to specify the structure of the goal term or certain transformations on the goal term.

A proof tactic which allows the user to transform a given goal into a new one, is the *rewrite tactic*. This tactic, based on Leibniz equality, was the tactic that I was tasked to implement.

A simple example of how this tactic functions would be as follows. Suppose we want to prove a goal G, which says that $(a + 0) + b = a + b$, for some $a, b \in \mathbb{N}$. Suppose, further, that we have already proved a lemma H saying that $\forall x \in \mathbb{N}, x + 0 = x$. The rewrite tactic would replace all occurrences of the first instance of the left-hand side (LHS) of H in G with the right-hand side (RHS) of H, giving us a new goal G', which would be $a + b = a + b$. We could then immediately prove this using reflexivity.

More formally, the setting we assume is as follows:

- We have a distinguished symbol for the binary predicate of equality which satisfies the Leibniz Axioms of equality. These axioms are:

  $$(EqRef.)\ \forall x : x = x.$$
  $$(EqInd.)\ \forall P, \forall x, y : x = y \Rightarrow (P(y) \Rightarrow P(x)).$$

  In Dedukti, we can define this symbol to be `eq` and encode[2] the axioms as:

[2]For more information on encodings in Dedukti see [4].

`symbol const eq` : $\forall a,\ T\ a \Rightarrow T\ a \Rightarrow Prop$

`symbol const refl` : $\forall a\ x,\ P\ (eq\ a\ x\ x)$

`symbol const eqind` : $\forall a\ x\ y,\ P\ (eq\ a\ x\ y) \Rightarrow \forall (p : T\ a \Rightarrow Prop),\ P\ (p\ y) \Rightarrow P\ (p\ x)$

- We have a lemma H which is an equality proof, of the form

$$\forall x_1, \ldots x_n : l(x_1, \ldots, x_n) = r(x_1, \ldots, x_n).$$

- We have a goal G, such that it contains at least one subterm that is is an instance of $l(x_1, \ldots, x_n)$. Suppose that the first subterm of G which can be unified with $l(x_1, \ldots, x_n)$ is $l(t_1, \ldots, t_n)$, where the t_i are terms.
- We want to produce a new goal G' such that all occurrences of the first instance of $l(x_1, \ldots, x_n)$ in G have been replaced by the corresponding instance of $r(x_1, \ldots, x_n)$, that is,

$$G' = G[r(t_1, \ldots, t_n)/l(t_1, \ldots, t_n].$$

- We want to specify that the new goal is now G' and to provide a mapping from proofs of G' to proofs of G.

The examples can get much more complicated than the one presented above, especially if a user requires only some occurrences of the first instance of the LHS of the lemma to be rewritten in the goal and not all. SSReflect is a Coq extension (used in the proof of the four colour theorem) which amongst other things allows user to select the subterms that the rewrite tactic will affect, using a fairly intuitive contextual pattern syntax. For more details see [5]. As the majority of Coq users are aware of SSReflect and know the syntax it was the syntax we chose to implement for the subterm selection mechanism in Dedukti.

From this point onward, as the appropriate terminology and motivation of my research has been established, I will summarise the placement in chronological order, assuming that the reader is familiar with the concepts already mentioned. I have made a conscious decision to avoid getting into a lot of theoretical details, but these can be found in the papers mentioned in the following section.

2.2 The First Few Weeks

On the first day of the Placement, after a short induction by Human Resources, I was given my office and a set of papers to read to get a broad overview of the work done by Deducteam and where my placement fits within this work. For further information, I include the list of papers below:

- Dedukti: A Logical Framework based on the λΠ-Calculus Modulo Theory [4]. This paper summarises the core work that Deducteam has done on Dedukti and it is essentially the main background necessary for understanding how the system works.

- Tactiques de preuve dans Dedukti [6]. This was a masters thesis, done in Deducteam, which covers the theoretical background for interactive proofs in the language of Dedukti.
- SSReflect: A Language of Patterns for Subterm Selection [7]. This paper was core for my placement, as my main task was to implement this language for subterm selection in Dedukti.

Throughout the first weeks Franck Slama, who was, amongst other things, also tasked with implementing proof tactics for Dedukti and who, as mentioned, co-supervised my placement gave me a series of lectures on Dependent Types and Automated Theorem Proving (the rough lines of the basic part of these lectures was presented in Section 2.1, above). At the same time, I was asked to start learning how to program in OCaml, the language in which Dedukti is implemented.

In the beginning of the second week, we held a meeting where a long term plan for the placement was established. The outline of the plan, presented below to give a broader overview of the placement, was the following:

- **June:** Literature review. This includes learning both Coq and OCaml, reading the papers above and understanding the theory on which the implementation will be based.
- **July (Early):** Determining the structure of the implementation and possible limitations
- **July (Late) - August (Early):** Implementation of the proof tactic for substitution of equal subterms in goals.
- **August (Late) - September (Early)**: Implementation of the subterm selection mechanism.
- **September (Late):** Cleaning up the implementation and producing some documentation. Preparation of a presentation for Deducteam.

I am proud to say that despite the difficulties that came up along the way, we managed to stay faithful to this plan.

2.3 A Small Scale Example

As at the beginning of the placement I was unfamiliar with OCaml, but once I was given sufficient time to practice we decided that it was time for me to implement a small example of how the subterm selection mechanism would work. This meant that I had to define a small abstract syntax tree for terms and patterns and attempt to implement the SSReflect selection mechanism in this simplified context.

This was extremely useful, as it allowed me to practice writing OCaml code, a lot of which was done in pair programming with Rodolphe Lepigre. As Rodolphe was the one who implemented most of the interactive theorem proving version of Dedukti, coding with him helped me learn the style of OCaml employed by the team and to answer any questions I had about OCaml. The OCaml implementation of this small project can be found in [8].

This toy-example can, in hindsight, be seen as an instance of how bite-sized tasks accumulate into something bigger. This was, in general, a common theme during the placement, as I was often

assigned seemingly easy tasks, that by themselves could be seen as meaningless but put together they constituted essentially of the puzzle pieces needed to construct the full picture. I feel, at this point, obligated to mention that if an attempt for a complete implementation was made from the beginning, something that I was inclined to do when starting out, I would not have succeeded in completing my placement objectives. To this end, the guidance I was given during the placement is much appreciated.

2.4 Implementing the Rewrite Tactic

One of the main deviations from the plan presented above was caused by the fact that the actual implementation of the simple rewrite tactic, with no patterns, was more complicated than expected.

Rodolphe Lepigre, one of the main developers of the newer version of Dedukti, which supports interactive theorem proving had co-developed an OCaml library which offers a set of tools for the manipulation of structures with bound and free variables called Bindlib, presented in [9]. This library lies in the heart of the new version of Dedukti, as the terms of the $\lambda\Pi$-calculus, in their internal representation are Bindlib binders.

The first step towards the implementation of a rewrite tactic was thus to get accustomed with Bindlib. Luckily, as I shared an office with its co-creator I had all the necessary support and managed to get my head around its subtleties.

The core of the simple rewrite tactic involves, as one may expect, a purely syntactic manipulation of terms in order to extract the necessary subterms and then reconstruct an updated term, in the sense described in Section 2.1. This proved to be more challenging than anticipated, simply because the high-level description of how rewrite is meant to function abstracts away all the complexity of handling terms in a purely syntactic and low-level manner.

Another challenge faced during this, initial, part of the implementation was the handling of metavariables, that is, objects that act as placeholders for terms that will be provided by the user in the future, whose internal representation is fairly complicated.

Combining, thus, both imperative and functional aspects the full implementation of the simple rewrite tactic was not completed as early in the placement as we had planned. That being said, in order to implement the basic rewrite I had to become very familiar with most of the back-end of the new version of Dedukti. That being said, the familiarity with the code-base that I gained during these weeks did in the end speed up the process of developing the rest of the tactic significantly.

2.5 The Last Few Weeks

Having, by the end of August, a fully working rewrite tactic, which subsumes the native Coq rewrite (with the exception of selecting subterms by indexing them with integers and conditional rewriting) it was time to commence the implementation of the full SSReflect syntax for subterm selection.

At this point of the placement as the ideas of patterns and subterm selection were often discussed in our meetings I had a clear view of how the implementation would proceed. Past this point, I could work independently, presenting my work to the people supervising me, only when I had achieved something.

In the course of two weeks, of fairly long hours of work, as the end of the placement was fast-

approaching, all of the SSReflect tactics were implemented. Finally, just before the end of the placement, a half hour presentation was given to the whole team, as most members were unfamiliar with the full implementation. In this presentation Georges Gonthier, one of the researchers responsible for the development of SSReflect was present, giving us the chance to have an interesting discussion on the subject.

The code for the implementation of rewrite with the SSReflect subterm selection mechanism can be found in [10]. It should be noted that this is the master branch of the new implementation of Dedukti and that any extensions of the tactic will be done on top of the code that was produced during the placement.

2.6 Organisation and Team Structure

As should be evident from the previous subsections, I was given a lot of support throughout the placement. This was mainly from my supervisor Frédéric Blanqui and the two post-doctoral members of the team, that are working on the new implementation of Dedukti, Rodolphe Lepigre and Franck Slama.

I met and discussed with Rodolphe on a daily basis, as we shared an office. Moreover, I met with Franck most of the days and at least once a week I met with all three to discuss my progress, potential pitfalls and next steps.

We also scheduled meetings every time I issued a pull request, to discuss implementation details more extensively. As the team works collaboratively, I had to be consistent with the style that they practice throughout the code-base. Moreover, the discussions and the comments on my code helped me improve both the style and the quality of the code I produced. This became evident, as the number of comments and corrections/suggestions that my pull requests received decreased significantly in the duration of the placement.

Apart from our meetings, Deducteam holds weekly meetings, where all aspects of Dedukti, from internal technicalities to how the syntax should be updated and everything in between are discussed by the entire team and each member presents their work and ideas. I was called to speak in one meeting when the rewrite tactic was first implemented and on another when the SSReflect syntax was completed.

Further to that, Deducteam holds weekly seminars, on Thursdays where team members or guest lecturers present some of their work. These seminars (held both in English and in French depending on the speaker and the audience) offered me a great chance to discover new and interesting topics in the forefront of theoretical computer science research. I was the speaker of the last Deducteam seminar during my placement, on the 27th of September.

In general, as the team is fairly small, everybody is kept in the loop, since besides the weekly stand-ups and meetings all of the members of the team go for lunch together and discuss issues during the day.

3 Imperial and the Placement

In this section I will be discussing how the courses offered by the Department of Computing and the Department of Maths were helpful during the placement. The focus of the section will be on outlining how the courses both supported and facilitated the research that I undertook.

It will be apparent that there is no mention to skills that I was missing at the outset of the placement, and this is for good reason. I feel like the first three years at Imperial had prepared me very well for the work that I was tasked to undertake.

3.1 Theoretical Background

The placement, as mentioned, started with a few weeks of preliminary reading on type theory, which felt like a direct continuation of the Type Systems for Programming Languages course offered in the third year. In a sense, this placement offered me a chance to delve deeper into subjects that Dr van Bakel alluded to, like Dependent Type Theory, the Curry-Howard correspondence and some specific type systems and formalisations (examples include the system F and of course the $\lambda\Pi$-calculus). Moreover, the chapter on Term Rewriting Systems in the course proved to be a good introduction for reading the first chapters of the book Term Rewriting and All That, which was very useful for understanding the Modulo Rewriting part of the $\lambda\Pi$-calculus Modulo Rewriting.

The Models of Computation course offered in the second year was a very good background for the placement. Notions like termination, confluence, normalisation etc., presented in a fairly simple context during the course were extremely useful to know. Moreover, both the courses mentioned above had large chapters on the λ-calculus and formal computation, including big-step semantics and typing derivations. These were core notions, without which I would have been unable to get past the preliminary papers that I had to read for the placement.

It should also be mentioned that the familiarity with symbolic logic, from both the first year Logic course offered by the Computing department and the third year Mathematical Logic course offered by the Maths department played an essential part in mentally preparing me for the levels of abstraction that research in type theory involves.

3.2 Practical Skills

The placement involved a lot of coding in OCaml an object-oriented functional programming language. Learning OCaml in a few weeks proved not to be as big a challenge as I had originally anticipated, since the first year course on Haskell gave a good background on functional programming. I should note here, that as practice to test my understanding of OCaml I implemented the Past Tests that Prof. Tony Field has made available on his website.

It goes without saying that the Java and C courses made writing imperative code fairly easy. The C course was especially useful, as there was a fair amount of pointer manipulation and side-effects involved in the implementation of the rewrite tactic.

The software engineering projects we had done were also a helpful background for working collaboratively, using version control. Thanks to these, working in a larger project with a number of collaborators proved not to be a difficult task. I should mention here that in my opinion the most helpful of the group projects was Pintos, as it was during Pintos that I learnt how to work in a large existing code-base, a skill that proved most valuable during the placement.

4 Major Benefits and Learning Outcomes

The aim of this section is to elaborate further on the positive experiences that I gained during the placement as well as to discuss the learning outcomes, both theoretical and practical, that I gained during the placement, mentioning skills that would have been useful to have beforehand.

4.1 Major Benefits

I opted to undertake a research placement instead of taking the standard industrial placement path that most students choose to follow. This was because I wish to, in the next year, pursue a PhD and I believed, prior to the placement, that working in research would be a good mental preparation for my PhD.

Although my role in the placement was essentially that of a "research engineer", that is, of a person that is mainly focused with implementational details, in a research oriented environment, I feel that I can confidently say that in this regard the placement was a success.

I did, indeed, get to experience how research in theoretical computer science is done in the real world, I got the chance to meet and exchange ideas with experts in the field and to discover modern areas of research in type theory[3].

Moreover, as I had to work both collaboratively and independently during the placement, I feel that I exercised both these skills significantly, and I consider that this improvement will eventually become increasingly useful as I enter the world of research.

4.2 Learning Outcomes

The main learning outcomes of the placement have been mentioned in passing throughout the previous sections, but as a summary, I will try to summarise them, in no particular order, below:

1. I practiced working collaboratively in a large, existing, code-base and I learned a new programming language.
2. I learned to work independently, with set deadlines and produce high-quality code.
3. I delved deeper into type theory and modern aspects of research in theoretical computer science.
4. I gained familiarity with proof assistants, both as an end-user and as a developer.
5. I got a first touch of how research is done at a higher level than that of an undergraduate degree.
6. I met and collaborated with a significant number of experts in the field of theoretical computer science.

In general, as should be evident from the list above (which is probably not exhaustive) I feel like I gained a lot from this placement and consider it a more than worthwile experience.

[3]I find it is necessary to mention that I do not aim to specialise in this field, but this does not mean that I did not find my learnings interesting.

5 Ethical and Professional Issues Encountered

In this section I will briefly discuss the ethical and professional issues that I encountered during my placement. This section is quite brief, as most of the ethical and professional issues one expects to encounter in industry are not especially relevant (and rarely come up) when working in research.

5.1 Respect for Other People

The lab, as part of the Ecole Normale Superieure is a multicultural environment where members are all aware and respectful of other cultures. People are not judged but on merit and the quality of their work. Apart from that, as would be the case in any computing department privacy and confidentiality are always respected.

5.2 Competence

The members of the team, be they researchers, engineers or human resource personnel are fairly accustomed to working with interns. They, as should be evident from Section 2 knew exactly how much time to spend working with me and how much time to give me to learn and work independently. As they became quickly aware of my skills they successfully assigned me tasks that are simultaneously challenging but not impossible, thus keeping me interested in the work throughout.

5.3 Integrity

The contributions of all members of the team are clearly reflected on git. Moreover, all current and past members of the team, including interns are listed on the website of the team. Moreover, as I had to work in a different location for a week, when the lab was closed for the summer, Inria covered my travelling arrangements.

Of course, one should be aware that when working for a Public Scientific and Technical Research Establishment there should be no conflicts of interest, that being said, even though all of the code on a public git repository and the work of the team is public record I was made to sign a non-disclosure agreement, to avoid potential conflicts of interest.

5.4 Environmental Issues

The LSV, where the team is located is part of the Ecole Normale Superieure Paris-Saclay, and as such it follows the environmental policies of the university, for instance there are recycle bins around the campus and members of staff as well as students are encouraged by the university to be environmentally aware.

6 Conclusion

In this section I will try to summarise the placement report and discuss my personal feelings on the placement.

My degree specialisation is on Pure Mathematics and Computational Logic and I think that in part this placement was a very useful part of the degree. As the research in which Deducteam focuses is close to my specialisation the placement did not feel detached from the rest of the courses at Imperial and was a valuable experience which I am glad to have had.

Moreover, I am very satisfied with the results of the placement, as the code that was produced over the past four months is now part of the newer version of Dedukti and will be maintained and hopefully extended in the future.

Overall, as should be evident from Section 4, I am very satisfied with the placement and grateful that I had the chance to have this experience.

References

[1] "Inria in Brief." `https://www.inria.fr/en/institute/inria-in-brief`. [Online; Accessed 07-October-2018].

[2] "Deducteam." `http://deducteam.gforge.inria.fr/`. [Online; Accessed 07-October-2018].

[3] W. Sandholtz and A. Sandholtz, *High-Tech Europe: The Politics of International Cooperation*. Comparative Studies of Health Systems and Medical Care, University of California Press, 1992. [Online, from Wikipedia; `https://en.wikipedia.org/wiki/Plan_Calcul`; Accessed 07-October-2018].

[4] A. Assaf, G. Burel, R. Cauderlierand, D. Delahaye, G. Dowek, C. Dubois, F. Gilbert, P. Halmagrand, O. Hermant, and R. Saillard", "Dedukti: a Logical Framework based on the $\lambda\Pi$-Calculus Modulo Theory." `http://www.lsv.fr/~dowek/Publi/expressing.pdf`. [Online; Accessed 07-October-2018].

[5] G. Gonthier, A. Mahboubi, and E. Tassi, "A Small Scale Reflection Extension for the Coq system," Research Report RR-6455, Inria Saclay Ile de France, 2016.

[6] A. Defourné, "Proof Tactics in Dedukti," Master's thesis, ENSIMAG, Sept. 2017.

[7] G. Gonthier and E. Tassi, "A language of patterns for subterm selection," in *ITP* (A. F. Lennart Beringer, ed.), vol. 7406 of *LNCS*, (Princeton, United States), pp. 361–376, Springer, Aug. 2012.

[8] A. Papadopoulos, "Term matching." `https://github.com/arisPapadop/term_matching`, 2018.

[9] R. Lepigre and C. Raffalli, "Abstract representation of binders in ocaml using the bindlib library," in Proceedings of the 13th International Workshop on *Logical Frameworks and Meta-Languages: Theory and Practice,* Oxford, UK, 7th July 2018 (F. Blanqui and G. Reis, eds.), vol. 274 of *Electronic Proceedings in Theoretical Computer Science*, pp. 42–56, Open Publishing Association, 2018.

[10] A. Papadopoulos, F. Slama, R. Lepigre, and F. Blanqui, "Rewrite tactic." `https://github.com/rlepigre/lambdapi/blob/master/src/rewrite.ml`, 2018.