This position's assignment is to extend and improve JSExplain to make it essential to the standardization process. We have thus developed a tool, JSExplain, to this end. Inria has been part of TC39 for 3 years, and we have witnessed a growing interest for tools to help the process.

In the setting of the ANR project Ajacs and the CominLabs project SecCloud, the goal is to develop and extend the JSEXplain tool, a JavaScript interpreter written in OCaml. JavaScript is a very complex language, containing many features that interact in a challenging way (the last version on the standard is more than 500 pages long). JavaScript is defined by a specification, called ECMAScript, that is designed by a standardization committee, called TC39. This committee includes many actors of the web: companies that build major browsers (Mozilla, Google, Apple, and Microsoft), companies that rely on JavaScript (Netflix, AirBnB, Paypal, …), and academic institutions (Inria, Imperial College London, …).

Adding features to JavaScript requires a careful study of the potential interactions with the existing features. This is crucial because the committee's motto is "We don't break the web". As soon as a feature is used in the wild, it cannot be removed from the standard.

To this end, the committee follows a strict process to evaluate proposals to extend the language. In particular, the last step of the standardization process requires that the feature be implemented in two major shipping browsers. As the implementations of browsers are complex, since they aim for high performance, it is difficult to experiment with variations of the feature during the development process. Inria has been part of TC39 for 3 years, and we have witnessed a growing interest for tools to help the standardization process. We have thus developed a tool, JSEXplain, to this end.

Our tool includes the following.

- An implementation of ECMAScript 5.1, written in a sub-language of OCaml and very close to the specification. The 5.1 version of the standard was shorter (200 pages long). JavaScript is defined by a specification, called ECMAScript, that is designed by a standardization committee, called TC39. This committee includes many actors of the web: companies that build major browsers (Mozilla, Google, Apple, and Microsoft), companies that rely on JavaScript (Netflix, AirBnB, Paypal, …), and academic institutions (Inria, Imperial College London, …).

- A compiler from the subset of OCaml to a subset of JavaScript, purely functional and typed. The compiled version of our interpreter consists of 17 000 lines of generated JavaScript.

- An instrumentation of the compilation generating traces during the execution of the program and a web-based visualizer of such traces. This enables a step-by-step interactive execution that shows both the state of the interpreted program and the state of the interpreter. A demo is available online (https://jscert.github.io/jsexplain/branch/master/driver.html).

- An integration with the test suite for JavaScript for both versions of the interpreter. A demo is available online (https://jscert.github.io/jsexplain/branch/master/driver.html).

Assignment
This position's assignment is to extend and improve JSExplain to make it essential to the standardization process. We have thus developed a tool, JSExplain, to this end. Inria has been part of TC39 for 3 years, and we have witnessed a growing interest for tools to help the process.

In the setting of the ANR project Ajacs and the CominLabs project SecCloud, the goal is to develop and extend the JSEXplain tool, a JavaScript interpreter written in OCaml. JavaScript is a very complex language, containing many features that interact in a challenging way (the last version on the standard is more than 500 pages long). JavaScript is defined by a specification, called ECMAScript, that is designed by a standardization committee, called TC39. This committee includes many actors of the web: companies that build major browsers (Mozilla, Google, Apple, and Microsoft), companies that rely on JavaScript (Netflix, AirBnB, Paypal, …), and academic institutions (Inria, Imperial College London, …).

Adding features to JavaScript requires a careful study of the potential interactions with the existing features. This is crucial because the committee's motto is "We don't break the web". As soon as a feature is used in the wild, it cannot be removed from the standard.

To this end, the committee follows a strict process to evaluate proposals to extend the language. In particular, the last step of the standardization process requires that the feature be implemented in two major shipping browsers. As the implementations of browsers are complex, since they aim for high performance, it is difficult to experiment with variations of the feature during the development process. Inria has been part of TC39 for 3 years, and we have witnessed a growing interest for tools to help the standardization process. We have thus developed a tool, JSEXplain, to this end.

Our tool includes the following.

- An implementation of ECMAScript 5.1, written in a sub-language of OCaml and very close to the specification. The 5.1 version of the standard was shorter (200 pages long). JavaScript is defined by a specification, called ECMAScript, that is designed by a standardization committee, called TC39. This committee includes many actors of the web: companies that build major browsers (Mozilla, Google, Apple, and Microsoft), companies that rely on JavaScript (Netflix, AirBnB, Paypal, …), and academic institutions (Inria, Imperial College London, …).

- A compiler from the subset of OCaml to a subset of JavaScript, purely functional and typed. The compiled version of our interpreter consists of 17 000 lines of generated JavaScript.

- An instrumentation of the compilation generating traces during the execution of the program and a web-based visualizer of such traces. This enables a step-by-step interactive execution that shows both the state of the interpreted program and the state of the interpreter. A demo is available online (https://jscert.github.io/jsexplain/branch/master/driver.html).

- An integration with the test suite for JavaScript for both versions of the interpreter.

Assignment
This position's assignment is to extend and improve JSExplain to make it essential to the standardization process. We have thus developed a tool, JSEXplain, to this end. Inria has been part of TC39 for 3 years, and we have witnessed a growing interest for tools to help the process.

In the setting of the ANR project Ajacs and the CominLabs project SecCloud, the goal is to develop and extend the JSEXplain tool, a JavaScript interpreter written in OCaml. JavaScript is a very complex language, containing many features that interact in a challenging way (the last version on the standard is more than 500 pages long). JavaScript is defined by a specification, called ECMAScript, that is designed by a standardization committee, called TC39. This committee includes many actors of the web: companies that build major browsers (Mozilla, Google, Apple, and Microsoft), companies that rely on JavaScript (Netflix, AirBnB, Paypal, …), and academic institutions (Inria, Imperial College London, …).

Adding features to JavaScript requires a careful study of the potential interactions with the existing features. This is crucial because the committee's motto is "We don't break the web". As soon as a feature is used in the wild, it cannot be removed from the standard.

To this end, the committee follows a strict process to evaluate proposals to extend the language. In particular, the last step of the standardization process requires that the feature be implemented in two major shipping browsers. As the implementations of browsers are complex, since they aim for high performance, it is difficult to experiment with variations of the feature during the development process. Inria has been part of TC39 for 3 years, and we have witnessed a growing interest for tools to help the standardization process. We have thus developed a tool, JSEXplain, to this end.

Our tool includes the following.

- An implementation of ECMAScript 5.1, written in a sub-language of OCaml and very close to the specification. The 5.1 version of the standard was shorter (200 pages long). JavaScript is defined by a specification, called ECMAScript, that is designed by a standardization committee, called TC39. This committee includes many actors of the web: companies that build major browsers (Mozilla, Google, Apple, and Microsoft), companies that rely on JavaScript (Netflix, AirBnB, Paypal, …), and academic institutions (Inria, Imperial College London, …).

- A compiler from the subset of OCaml to a subset of JavaScript, purely functional and typed. The compiled version of our interpreter consists of 17 000 lines of generated JavaScript.

- An instrumentation of the compilation generating traces during the execution of the program and a web-based visualizer of such traces. This enables a step-by-step interactive execution that shows both the state of the interpreted program and the state of the interpreter. A demo is available online (https://jscert.github.io/jsexplain/branch/master/driver.html).

- An integration with the test suite for JavaScript for both versions of the interpreter.
standardization process.

Our mid-term goal is the endorsement by TC39 either of this implementation as a reference interpreter, or of a similar version in a well specified language compiled to a reference interpreter. The well-specified language could for instance be an extension of ECMArkup, which is currently used to describe the algorithms of the specification. In either case, we want to be able to easily follow the evolutions of the specification. In the long term, we want to provide tools to enable the committee to update the reference interpreter by itself.

The recruited person will participate to some TC39 meetings. Travel expenses will be reimbursed up to the regulatory limits.

Main activities

- Extension of our translation tool from OCaml to JavaScript
- Study of the specification and extension of its implementation in OCaml to cover the whole standard
- Improvement of the interactive visualization tool, in particular by adding links to the online specification
- Involvement in the standardization process and presentation of the tool to the standardization committee

Skills

- Functional programming in OCaml
- Writing reports and presenting results
- JavaScript experience is not required but is a plus

Benefits package

- Subsidised catering service
- Partially-reimbursed public transport
- Social security
- Paid leave
- Flexible working hours
- Sports facilities

Remuneration

A partir de 2 562€ brut mensuel