Privilege separation in browser architectures

Michele Bugliesi   Stefano Calzavara   Enrico Steffinlongo

June 2, 2014

Abstract

In many software systems as modern web browsers the user and his sensitive data often interact with the untrusted outer world. This scenario can pose a serious threat to the user's private data and gives new relevance to an old story in computer science: providing controlled access to untrusted components, while preserving usability and ease of interaction. To address the threats of untrusted components, modern web browsers propose privilege-separated architectures, which isolate components that manage critical tasks and data from components which handle untrusted inputs. The former components are given strong permissions, possibly coinciding with the full set of permissions granted to the user, while the untrusted components are granted only limited privileges, to limit possible malicious behaviours: all the interactions between trusted and untrusted components is handled via message passing. In this thesis we introduce a formal semantics for privilege-separated architectures and we provide a general definition of privilege separation: we discuss how different privilege-separated architectures can be evaluated in our framework, identifying how different security threats can be avoided, mitigated or disregarded. Specifically, we evaluate in detail the existing Google Chrome Extension Architecture in our formal model and we discuss how its design can mitigate serious security risks, with only limited impact on the user experience.

Contents

Chapter 1

Motivation

1.1 Privilege separation

1.2 Privilege escalation attacks

1.3 Chrome extension architecture overview

Chrome by Google, as all actual-days browsers, provides a powerful extension framework. This gives to developers a huge architecture made explicitly to extend the core browser potentiality in order to build small programs that enhance user-experience. In Chrome web store there are lot of extensions with very various behaviors like security enhancers, theme changers, organizers or other utilities, multimedia visualizer, games and others. For example, AdBlock (one of the top downloaded) is an extension made to block all ads on websites; ShareMeNot "protects the user against being tracked from third-party social media buttons while still allowing it to use them"[3]. As we can notice extensions have different purposes, and many of them has to interact massively with web pages. This creates a very large attack surface for attackers and is a big threat for the user. Moreover many extensions are written by developers that are not security experts so, even if their behavior is not malign, the bugs that can appear in them can be easily exploited by attackers.

To mitigate this threat, as deeply discussed in [4], the extension framework is built to force programmers to adopt privilege separation, least privilege and strong isolation. Privilege separation, as explained before in 1.1, force the developer to split the application in components providing for the communication a message passing interface; least privilege gives to the app the least set of permission needed through the execution of the extension and the strong isolation separate the heaps of the various components of the extension running them in different processes in order to block any possible escalation and direct delegation.

More specifically, Google Chrome extension framework [2] splits the extension in two sets components: content scripts and background pages. The content scripts are injected in every page on which the extension is running using the same origin; they run with no privileges except the one used to send messages to the background and they cannot exchange pointers with the page except to the standard field of the DOM. Background pages, instead, have only one instance for each extension, are totally separated from the opened pages, have the full set of privilege granted at install time and, if it is allowed from the manifest, they can inject new content scripts to pages, but they can communicate with the content scripts only via message passing.

1.4 Chrome extension architecture weaknesses

1.5 Proposal

In this work do a study on Chrome Extensions identifying a possible weakness. We write a calcolus

Chapter 2

Background

2.1 Chrome extension architecture details

As showed in [2] a Chrome Extension is an archive containing files of various kind like JavaScript, HTML, JSON, pages, images and other that extends the browser features.

A basic extension contains a manifest file and one or more Javascript or Html files.

2.1.1 Manifest

The manifest file `manifest.json` is a JSON-formatted file that with all the specification of the extension. It is the entry point of the extension and contain two mandatory fields: `name` and `version` containing the name and the version of the extension. Other important field are `background`, `content_scripts`, `permissions` and we will explain here.

- `background`: has or a `script` field containing the source of the content script or a `page` field containing the source of an HTML page. If the `script` field is used the scripts are injected in a empty extension core page, while if is used `page` the HTML document with all his elements, including scripts, compose the extension core.

- `content_scripts`: contains a list of content script objects. A content script object can contains field `js` that contain the list of Javascript files to be injected and other, and must contain field `matches`: a list of match patterns. Match patterns are explained below.

- `permissions`: contains a list of privileges that are requested by the extension. These can be either a host match pattern for XHR request to that host or the name of the API needed.

Another possible field is `optional_permissions`. It contains the list of optional permission that the extension could require and are used to restrict the privilege granted to the app. To use one of this permissions the background page has to require explicitly them and to release after use. Using the optional permission is possible to reduce the possible privileges escalated by an attacker, **but are used rarely and are not in our interest**. Togliere?

Table 2.1 Url pattern syntax. Table taken from [1]

```
<url-pattern> := <scheme>://<host><path>
<scheme> := '*' | 'http' | 'https' | 'file' | 'ftp' | 'chrome-extension'
<host> := '*' | '*.' <any char except '/' and '*'>+
<path> := '/' <any chars>
```

Table 2.2 A manifest file

```
{
  "manifest_version": 2,
  "name":"Moodle expander",
  "description":"Download homework and uploads marks from a JSON
     string",
  "version":"1",
  "background": { "scripts": ["background.js"] },
  "permissions":
    [
      "tabs",
      "downloads",
      "https://moodle.dsi.unive.it/*"
    ],
  "content_scripts":
    [
      {
        "matches": ["https://moodle.dsi.unive.it/*"],
        "js": ["myscript.js"]
      }
    ]
}
```

A match pattern is a string composed of three parts: `scheme`, `host` and `path`. A part can contains a value or `"*"` that means all possible values. In table 2.1 is shown the syntax of the URL patterns. For more details refer to [1]. As we can see we can decide to inject some content scripts on pages derived from a given match. This is used when the extension has to interact with only certain pages. For example `"*://*/*"` means all pages; `"https://*/*"` means all HTTPS pages; `"https://*.google.com/*"` means all HTTPS pages with google as host and with all path (e.g., `mail.google.com`, `www.google.com`, `docs.google.com/mine`).

In table 2.2 we can see a manifest of a simple Chrome extension that expands the feature of moodle. We can see that the extension has an empty background page on which is injected the file `background.js` an that has tabs, downloads permission and that can execute XHR to all path contained in `https://moodle.dsi.unive.it/`. It has also one content script that is injected in all subpages of `https://moodle.dsi.unive.it/`.

2.1.2 Content script

Content script are Javascript source files that are automatically injected to the web page if this match with the pattern defined in the manifest. In the example of table 2.1 the file `myscript.js` is injected to all sub-pages of `https://moodle.dsi.unive.it/`. In the extension framework content scripts are designed to interact with the page. Since this interaction could be the entry point for an attacker, content scripts have no permissions except the one used to communicate with the extension core. In order to reduce injection of code in the content script from a malign page, there is a strong isolation between the heaps of these two. Content scripts of same extension are run together in their own address space, and the only way they have to interact with the page is via DOM API. As explained in [4] browser provide one common DOM element accessed via its API and all scripts both on the page or in the extension can modify it, but only changes of the standard DOM properties are shared, while other changes are kept locally.

mettere figur dell'articolo?

The message passing interface has crucial importance in this work since it is the only way for a content script to trigger execution of a privilege. We will discuss it later in 2.1.4.

2.1.3 Extension core

The extension core is the most critical part of the application. It is executed in a unique origin like `chrome-extension://hcdmlbjlcojpbbinplfgbjodclfijhce` in order to prevent cross origin attacks, but can communicate with all origins that match with one of the host permission requested. In this environment are executed all scripts defined in the background field of the manifest. Since background pages can have remote object, they can also request to the web such resources, but this can be very dangerous because if the resources are on simple HTTP connections them can be altered by an attacker. In [5] is described how to enforce the security policy in order to avoid such possible weakness. As already said background pages can interact with content scripts via message passing.

2.1.4 Message passing API

Every content script of the extension can access `chrome.runtime` that is the object on wich the message passing interface is implemented(for more details refer to [?]).

The main method to send a message to the extension core is invoking the method `chrome.runtime.sendMessage`. Like all Chrome APIs even the message passing is asynchronous. As primary arguments it takes the message that can be of any kind and a callback function that is triggered if someone answer to the message. The message, before sending is marshaled using a JSON serializer. This prevent exchange of pointers or of functions, but limits the expressiveness of the prototype-based object-oriented feature of Javascript. It also fails in presence of recursive objects.

An element, to listen to inbound messages, has to register a function on the `chrome.runtime.onMessa` event. This function will be triggered when a message arrives. Its arguments are the message (unmarshalled by the API), the sender and an optional callback used to send response to that message. The sender field is very important because is the only warranty about the sender. In fact the message may not be used to decide the sender, because it can be of every kind.

Table 2.3 Sending a message.

Sender

```
var info = "hello";
var callback =
  function(response)
  {
    console.log("get response
        : " + response);
  };
chrome.runtime.sendMessage(
   info, callback);
```

Receiver

```
var onMessage =
  function(message, sender,
     sendResponse)
  {
    if (message = "hello")
    {
        //compute message
      sendResponse("hi");
    }
    else
      console.log("connection
          refused from"+
         sender);
  };
chrome.runtime.onMessage.
   addListener(onMessage);
```

Since content scripts are multiple and injected in various pages (tabs), the extension core for sending a message has to use the `sendMessage` method proper of the `tab` object to which the message has to be sent. Its behavior is the same of the `chrome.runtime.sendMessage` method.

In table 2.3 we can see how to use the simple message passing interface. A component simply sends the message and wait for a response. The other register `onMessage` function as event listener for messages. When it is triggered by an incoming message `onMessage` check the message and decide to compute something with the request or to refuse the message. Unfortunately this practice is very dangerous because there are no warranties on the source of the message inside it, so an attacker can sends false messages that trigger security sensitive computation with untrusted source.

Another way to communicate, that is more secure, is done using a channel as in table 2.4. In the message passing API there is a method called `connect` that takes as optional arguments a message to deliver when the corresponding event `onConnect` is triggered and that returns a port. Such object is a bidirectional channel that can be used to communicate and contains the methods `postMessage`, `disconnect` and the events `onMessage` and `onDisconnect`. Communication using ports instead of the classical `chrome.runtime.sendMessage` is more secure, because only who has the port endpoints can communicate. This grant the sender of the message.

2.2 Flow logic

Table 2.4 Port creation.

Port opening active	Port opening passive

<pre>var port = chrome.runtime.
   connect({name: "cs1"});
port.onMessage.addListener(
   onMessage)
port.postMessage("hi")</pre>
	<pre>var scriptPort = null;
	var onConnect =
	  function(port)
	  {
	    if (port.name = "cs1")
	    {
	      scriptPort = port;
	      port.onMessage.
	         addListener(
	         onMessage);
	    }
	    else
	    {
	      console.log("connection
	          refused");
	      port.disconnect();
	    }
	  };
	chrome.runtime.onConnect.
	   addListener(onConnect)</pre>

Chapter 3

Formalization

3.1 Threat Model

3.2 Calculus

3.3 Safety properties

3.4 Analysis specification

3.4.1 Abstract succinct

$$
\begin{array}{lrcl}
\textit{Abstract cache} & \hat{C} & : & \mathcal{L} \rightarrow \hat{V} \\
\textit{Abstract variable environment} & \hat{\Gamma} & : & \mathcal{V} \rightarrow \hat{V} \\
\textit{Abstract memory} & \hat{\mu} & : & \mathcal{L} \times \mathcal{P} \rightarrow \hat{V} \\
\textit{Abstract permission cache} & \hat{P} & : & \mathcal{L} \rightarrow \mathcal{P}
\end{array}
$$

$$
\begin{array}{ll}
[\mathit{PE\text{-}Val}] & (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} c : \hat{v} \text{ iff } \{d_c\} \subseteq \hat{v} \\
[\mathit{PE\text{-}Var}] & (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} x : \hat{v} \text{ iff } \hat{\Gamma}(x) \subseteq \hat{v} \\
[\mathit{PE\text{-}Lambda}] & (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} \lambda x.e : \hat{v} \text{ iff } \{\lambda x.e\} \subseteq \hat{v} \\
[\mathit{PE\text{-}Obj}] & (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} \{\overrightarrow{str_i : e_i}\} : \hat{v} \gg \rho \text{ iff} \\
& \quad \forall i : (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_i : \hat{v}_i \gg \rho_i \wedge \\
& \qquad \{\overrightarrow{str_i : \hat{v}_i}\} \subseteq \hat{v} \wedge \\
& \qquad \rho_i \sqsubseteq \rho \\
[\mathit{PE\text{-}Let}] & (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} \textbf{let } \overrightarrow{x_i = e_i} \textbf{ in } e' : \hat{v} \gg \rho \text{ iff} \\
& \quad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e' : \hat{v} \gg \rho' \wedge \\
& \quad \rho' \sqsubseteq \rho \wedge \\
& \quad \forall i : \\
& \qquad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_i : \hat{v}_i \gg \rho_i \wedge \\
& \qquad \hat{v}_i \subseteq \hat{\Gamma}(x_i) \wedge \\
& \qquad \rho_i \sqsubseteq \rho \\
[\mathit{PE\text{-}App}] & (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_1\, e_2 : \hat{v} \gg \rho \text{ iff} \\
& \quad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_1 : \hat{v}_1 \gg \rho_1 \wedge \\
& \quad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_2 : \hat{v}_2 \gg \rho_2 \wedge \\
& \quad \rho_1 \sqsubseteq \rho \wedge \\
& \quad \rho_2 \sqsubseteq \rho \wedge \\
& \quad \forall (\lambda x.e_0) \in \hat{v}_1 : \\
& \qquad \hat{v}_2 \subseteq \hat{\Gamma}(x) \wedge \\
& \qquad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_0 : \hat{v}_0 \gg \rho_0 \wedge \\
& \qquad \rho_0 \sqsubseteq \rho \wedge \\
& \qquad \hat{v}_0 \subseteq \hat{v} \\
[\mathit{PE\text{-}Op}] & (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} op(\overrightarrow{e_i}) : \hat{v} \gg \rho \text{ iff} \\
& \quad \forall i : \\
& \qquad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_i : \hat{v}_i \gg \rho_i \wedge \\
& \qquad \rho_i \sqsubseteq \rho \wedge \\
& \quad \widehat{op}(\overrightarrow{\hat{v}_i}) \subseteq \hat{v} \\
[\mathit{PE\text{-}Cond}] & (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} \textbf{if } (e_0)\ \{\ e_1\ \} \textbf{ else } \{\ e_2\ \} : \hat{v} \gg \rho \text{ iff} \\
& \quad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_0 : \hat{v}_0 \gg \rho_0 \wedge \\
& \quad \rho_0 \sqsubseteq \rho \wedge \\
& \quad \widehat{\textbf{true}} \in \hat{v}_0 \Rightarrow \\
& \qquad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_1 : \hat{v}_1 \gg \rho_1 \wedge \hat{v}_1 \subseteq \hat{v} \wedge \rho_1 \sqsubseteq \rho \wedge \\
& \quad \widehat{\textbf{false}} \in \hat{v}_0 \Rightarrow \\
& \qquad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_2 : \hat{v}_2 \gg \rho_2 \wedge \hat{v}_2 \subseteq \hat{v} \wedge \rho_2 \sqsubseteq \rho \\
[\mathit{PE\text{-}While}] & (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} \textbf{while } (e_1)\ \{\ e_2\ \} : \hat{v} \gg \rho \text{ iff} \\
& \quad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_1 : \hat{v}_1 \gg \rho_1 \wedge \\
& \quad \rho_1 \sqsubseteq \rho \wedge \\
& \quad \widehat{\textbf{true}} \in \hat{v}_1 \Rightarrow \\
& \qquad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_2 : \hat{v}_2 \gg \rho_2 \wedge \hat{v}_2 \subseteq \hat{v} \wedge \rho_2 \sqsubseteq \rho \wedge \\
& \quad \widehat{\textbf{false}} \in \hat{v}_1 \Rightarrow \\
& \qquad \widehat{\textbf{undefined}} \subseteq \hat{v} \\
[\mathit{PE\text{-}GetField}] & (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_1[e_2] : \hat{v} \gg \rho \text{ iff} \\
& \quad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_1 : \hat{v}_1 \gg \rho_1 \wedge \\
& \quad \rho_1 \sqsubseteq \rho \wedge \\
& \quad (\hat{\Gamma}, \hat{\mu}) \models_{\rho_s} e_2 : \hat{v}_2 \gg \rho_2 \wedge \\
& \quad \rho_2 \sqsubseteq \rho \wedge \\
& \quad \widehat{get}(\hat{v}_1, \hat{v}_2) \subseteq \hat{v}
\end{array}
$$

$$
\begin{array}{ll}
[\textit{PE-SetField}] & (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e_0[e_1] = e2 : \hat{v} \gg \rho \text{ iff} \\
& \quad (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e_0 : \hat{v}_0 \gg \rho_0 \wedge \\
& \quad \rho_0 \sqsubseteq \rho \wedge \\
& \quad (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e_1 : \hat{v}_1 \gg \rho_1 \wedge \\
& \quad \rho_1 \sqsubseteq \rho \wedge \\
& \quad (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e_2 : \hat{v}_2 \gg \rho_2 \wedge \\
& \quad \rho_2 \sqsubseteq \rho \wedge \\
& \quad \widehat{set}(\hat{v}_0,\hat{v}_1,\hat{v}_2) \subseteq \hat{v} \\
[\textit{PE-DelField}] & (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} \textbf{delete } e_1[e_2] : \hat{v} \gg \rho \text{ iff} \\
& \quad (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e_1 : \hat{v}_1 \gg \rho_1 \wedge \\
& \quad \rho_1 \sqsubseteq \rho \wedge \\
& \quad (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e_2 : \hat{v}_2 \gg \rho_2 \wedge \\
& \quad \rho_2 \sqsubseteq \rho \wedge \\
& \quad \widehat{del}(\hat{v}_1,\hat{v}_2) \subseteq \hat{v} \\
[\textit{PE-Ref}] & (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} \textbf{ref}_{r,\rho_r}\ e : \{r\} \gg \rho \text{ iff} \\
& \quad (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e : \hat{v} \gg \rho \wedge \\
& \quad \rho_r \sqsubseteq \rho_s \Rightarrow \hat{v} \subseteq \hat{\mu}(r,\rho_r) \\
[\textit{PE-DeRef}] & (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} \textbf{deref } e : \hat{v} \gg \rho \text{ iff} \\
& \quad (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e : \hat{v}_1 \gg \rho_1 \wedge \\
& \quad \rho_1 \sqsubseteq \rho \wedge \\
& \quad \forall r \in \hat{v}_1 : \forall \rho_r \sqsubseteq \rho_s : \hat{\mu}(r,\rho_r) \subseteq \hat{v} \\
[\textit{PE-SetRef}] & (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e_1 = e_2 : \hat{v} \gg \rho \text{ iff} \\
& \quad (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e : \hat{v}_1 \gg \rho_1 \wedge \\
& \quad \rho_1 \sqsubseteq \rho \wedge \\
& \quad (\hat{\Gamma},\hat{\mu}) \models_{\rho_s} e_2 : \hat{v}_2 \gg \rho_2 \wedge \\
& \quad \rho_2 \sqsubseteq \rho \wedge \\
& \quad \forall r \in \hat{v}_1 : \forall \rho_r \sqsubseteq \rho_s : \\
& \qquad \hat{v}_2 \subseteq \hat{\mu}(r,\rho_r) \wedge \\
& \qquad \hat{v}_2 \subseteq \hat{v} \\
[\textit{PE-Send}] & \ldots \\
[\textit{PE-Err}] & \ldots \\
[\textit{PE-Exercise}] & \ldots
\end{array}
$$

3.4.2 Compositional Verbose

$$
\begin{array}{ll}
[\mathit{CV\text{-}Val}] & (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} (c)^{\ell} \text{ iff } \{d_c\} \subseteq \hat{C}(\ell) \\
[\mathit{CV\text{-}Var}] & (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} (x)^{\ell} \text{ iff } \hat{\Gamma}(x) \subseteq \hat{C}(\ell) \\
[\mathit{CV\text{-}Lambda}] & (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} (\lambda x.{e_0}^{\ell_0})^{\ell} \text{ iff} \\
& \quad \{\lambda x.{e_0}^{\ell_0}\} \subseteq \hat{C}(\ell) \wedge \\
& \quad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_0}^{\ell_0} \\
[\mathit{CV\text{-}Obj}] & (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} (\{\overrightarrow{str_i : {e_i}^{\ell_i}}\})^{\ell} \text{ iff} \\
& \quad \forall i : \\
& \qquad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_i}^{\ell_i} \wedge \\
& \qquad \hat{P}(\ell_i) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \{\overrightarrow{str_i : \hat{C}(\ell_i)}\} \subseteq \hat{C}_{\ell} \\
[\mathit{CV\text{-}Let}] & (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} (\textbf{let } \overrightarrow{x_i = {e_i}^{\ell_i}} \textbf{ in } {e'}^{\ell'})^{\ell} \text{ iff} \\
& \quad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e'}^{\ell'} \wedge \\
& \quad \hat{P}(\ell') \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \hat{C}(\ell') \subseteq \hat{C}(\ell) \wedge \\
& \quad \forall i : \\
& \qquad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_i}^{\ell_i} \wedge \\
& \qquad \hat{C}(\ell_i) \subseteq \hat{\Gamma}(x_i) \wedge \\
& \qquad \hat{P}(\ell_i) \sqsubseteq \hat{P}(\ell) \\
[\mathit{CV\text{-}App}] & (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} ({e_1}^{\ell_1}\, {e_2}^{\ell_2})^{\ell} \text{ iff} \\
& \quad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_1}^{\ell_1} \wedge (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_2}^{\ell_2} \wedge \\
& \quad \hat{P}(\ell_1) \sqsubseteq \hat{P}(\ell) \wedge \hat{P}(\ell_2) \sqsubseteq \hat{P}(\ell) \\
& \quad \forall (\lambda x.{e_0}^{\ell_0}) \in \hat{C}(\ell_1) : \\
& \qquad \hat{C}(\ell_2) \subseteq \hat{\Gamma}(x) \wedge \hat{C}(\ell_0) \subseteq \hat{C}(\ell) \wedge \\
& \qquad \hat{P}(\ell_0) \sqsubseteq \hat{P}(\ell) \\
[\mathit{CV\text{-}Op}] & (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} (op(\overrightarrow{{e_i}^{\ell_i}}))^{\ell} \text{ iff} \\
& \quad \forall i : \\
& \qquad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_i}^{\ell_i} \wedge \\
& \qquad \hat{P}(\ell_i) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \widehat{op}(\hat{C}(\ell_i)) \subseteq \hat{C}(\ell) \\
[\mathit{CV\text{-}Cond}] & (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} (\textbf{if } ({e_0}^{\ell_0}) \; \{ \; {e_1}^{\ell_1} \; \} \textbf{ else } \{ \; {e_2}^{\ell_2} \; \})^{\ell} \text{ iff} \\
& \quad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_0}^{\ell_0} \wedge \\
& \quad \hat{P}(\ell_0) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \widehat{\textbf{true}} \in \hat{C}(\ell_0) \Rightarrow \\
& \qquad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_1}^{\ell_1} \wedge \hat{C}(\ell_1) \subseteq \hat{C}(\ell) \wedge \\
& \qquad \hat{P}(\ell_1) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \widehat{\textbf{false}} \in \hat{C}(\ell_0) \Rightarrow \\
& \qquad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_2}^{\ell_2} \wedge \hat{C}(\ell_2) \subseteq \hat{C}(\ell) \wedge \\
& \qquad \hat{P}(\ell_2) \sqsubseteq \hat{P}(\ell) \\
[\mathit{CV\text{-}While}] & (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} (\textbf{while } ({e_1}^{\ell_1}) \; \{ \; {e_2}^{\ell_2} \; \})^{\ell} \text{ iff} \\
& \quad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_1}^{\ell_1} \wedge \\
& \quad \hat{P}(\ell_1) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \widehat{\textbf{true}} \in \hat{C}(\ell_1) \Rightarrow \\
& \qquad (\hat{C}, \hat{\Gamma}, \hat{\mu}, \hat{P}) \models_{c\rho_s} {e_2}^{\ell_2} \wedge \hat{C}(\ell_2) \subseteq \hat{C}(\ell) \wedge \\
& \qquad \hat{P}(\ell_2) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \widehat{\textbf{false}} \in \hat{C}(\ell_1) \Rightarrow \widehat{\textbf{undefined}} \subseteq \hat{C}(\ell)
\end{array}
$$

$$
\begin{array}{ll}
[\mathit{CV\text{-}GetField}] & (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} ({e_1}^{\ell_1}[{e_2}^{\ell_2}])^{\ell} \text{ iff} \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_1}^{\ell_1} \wedge \\
& \quad \hat{P}(\ell_1) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_2}^{\ell_2} \wedge \\
& \quad \hat{P}(\ell_2) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \widehat{get}(\hat{C}(\ell_1),\hat{C}(\ell_2)) \subseteq \hat{C}(\ell) \\
[\mathit{CV\text{-}SetField}] & (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} ({e_0}^{\ell_0}[{e_1}^{\ell_1}] = {e_2}^{\ell_2})^{\ell} \text{ iff} \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_0}^{\ell_0} \wedge \\
& \quad \hat{P}(\ell_0) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_1}^{\ell_1} \wedge \\
& \quad \hat{P}(\ell_1) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_2}^{\ell_2} \wedge \\
& \quad \hat{P}(\ell_2) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \widehat{set}(\hat{C}(\ell_0),\hat{C}(\ell_1),\hat{C}(\ell_2)) \subseteq \hat{C}(\ell) \\
[\mathit{CV\text{-}DelField}] & (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} (\mathbf{delete}\ {e_1}^{\ell_1}[{e_2}^{\ell_2}])^{\ell} \text{ iff} \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_1}^{\ell_1} \wedge \\
& \quad \hat{P}(\ell_1) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_2}^{\ell_2} \wedge \\
& \quad \hat{P}(\ell_2) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \widehat{del}(\hat{C}(\ell_1),\hat{C}(\ell_2)) \subseteq \hat{C}(\ell) \\
[\mathit{CV\text{-}Ref}] & (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} (\mathbf{ref}_{r,\rho_r}\ {e_1}^{\ell_1})^{\ell} \text{ iff} \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_1}^{\ell_1} \wedge \\
& \quad \{r\} \subseteq \hat{C}(\ell) \wedge \\
& \quad \hat{P}(\ell_1) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \rho_r \sqsubseteq \rho_s \Rightarrow \hat{C}(\ell_1) \subseteq \hat{\mu}(r,\rho_r) \\
[\mathit{CV\text{-}DeRef}] & (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} (\mathbf{deref}\ {e_1}^{\ell_1})^{\ell} \text{ iff} \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_1}^{\ell_1} \wedge \\
& \quad \hat{P}(\ell_1) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \forall r \in \hat{C}(\ell_1) : \forall \rho_r \sqsubseteq \rho_s : \\
& \quad\quad \hat{\mu}(r,\rho_r) \subseteq \hat{C}(\ell) \\
[\mathit{CV\text{-}SetRef}] & (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} ({e_1}^{\ell_1} = {e_2}^{\ell_2})^{\ell} \text{ iff} \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_1}^{\ell_1} \wedge \\
& \quad \hat{P}(\ell_1) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad (\hat{C},\hat{\Gamma},\hat{\mu},\hat{P}) \models_{c\rho_s} {e_2}^{\ell_2} \wedge \\
& \quad \hat{P}(\ell_2) \sqsubseteq \hat{P}(\ell) \wedge \\
& \quad \forall r \in \hat{C}(\ell_1) : \forall \rho_r \sqsubseteq \rho_s : \\
& \quad\quad \hat{C}(\ell_2) \subseteq \hat{\mu}(r,\rho_r) \wedge \\
& \quad \hat{C}(\ell_2) \subseteq \hat{C}(\ell) \\
[\mathit{PE\text{-}Send}] & \ldots \\
[\mathit{PE\text{-}Err}] & \ldots \\
[\mathit{PE\text{-}Exercise}] & \ldots
\end{array}
$$

3.5 Theorem

3.6 Requirements for correctness

Chapter 4

Abstract Domains

4.1 Abstract domains choice

$R_1 = \{\overrightarrow{\widehat{str_i} : \widehat{v_i}}\} \sqsubseteq \{\overrightarrow{\widehat{str_j} : \widehat{v_j}}\} = R_2$ sse:

1. R_1 ha meno campi di R_2
2. ogni campo di R_1 e' piu' preciso del **corrispondente** campo di R_2

$\forall i, \exists j : \widehat{str_i} \sqsubseteq \widehat{str_j}$
$\forall i, \exists j : \widehat{str_i} \sqsubseteq \widehat{str_j} \Rightarrow \widehat{v_i} \sqsubseteq \widehat{v_j}$
Set:

- Exact
  - $\exists \rightarrow Union$
  - $\nexists \rightarrow addinprefix$
- Prefix
  - aggiungo in $*$

$\hat{v} \sqsubseteq \hat{v}'$ iff $\forall \widehat{u}_i \in \hat{v}, \exists \widehat{u}_j \in \hat{v}' : \widehat{u}_i \sqsubseteq \widehat{u}_j$.
If Galois connection then
$\hat{v} \sqsubseteq \hat{v}'$ iff $\gamma(\hat{v}) \subseteq \gamma(\hat{v}')$
where $\gamma : \widehat{V} \rightarrow P(V)$ is the concretisation function.
$\gamma_p : \widehat{PV} \rightarrow P(V)$
$\gamma(\hat{v}) = \bigcup_{\widehat{u}_i \in \hat{v}} \gamma_p(\widehat{u}_i)$

$\widehat{pre_{bool}} = \widehat{true}|\widehat{false}$

$\widehat{u_{bool}} = \{\overrightarrow{\widehat{pre_{bool}}}\}$ with $\sqsubseteq = \subseteq$

$\widehat{pre_{int}} = \oplus|0|\ominus$

$\widehat{u_{int}} = \{\overrightarrow{\widehat{pre_{int}}}\}$ with $\sqsubseteq = \subseteq$

$\widehat{pre_{string}} = s|s*$

$\widehat{u_{string}} = \{\overrightarrow{\widehat{pre_{string}}}\}$ with $\sqsubseteq = \subseteq$
— Giulia's spec. is more tricky than $\subseteq$

$\widehat{pre_{ref}} = r$

$\widehat{u_{ref}} = \{\overrightarrow{\widehat{pre_{ref}}}\}$ with $\sqsubseteq = \subseteq$

$\widehat{pre_{\lambda}} = \lambda$

$\widehat{u_{\lambda}} = \{\overrightarrow{\widehat{pre_{\lambda}}}\}$ with $\sqsubseteq = \subseteq$

$\widehat{pre_{rec}} = \{\overrightarrow{\widehat{str_i} : \hat{v}_i}\}$

$\widehat{u_{rec}} = \widehat{pre_{rec}}$ with $\sqsubseteq = \widehat{u_{rec}}_{\sqsubseteq}$

$\hat{v} = (\widehat{u_{bool}}, \widehat{u_{int}}, \widehat{u_{string}}, \widehat{u_{ref}}, \widehat{u_{\lambda}}, \widehat{u_{rec}}, \{\widehat{Null}\}, \{\widehat{Undef}\})$

with $\hat{v} \sqsubseteq \hat{v}'$ iff
$\widehat{u_{bool}} \sqsubseteq \widehat{u_{bool}}' \wedge$
$\widehat{u_{int}} \sqsubseteq \widehat{u_{int}}' \wedge$
$\widehat{u_{string}} \sqsubseteq \widehat{u_{string}}' \wedge$
$\widehat{u_{ref}} \sqsubseteq \widehat{u_{ref}}' \wedge$
$\widehat{u_{\lambda}} \sqsubseteq \widehat{u_{\lambda}}' \wedge$
$\widehat{u_{rec}} \sqsubseteq \widehat{u_{rec}}' \wedge$
$\widehat{Null} \notin \hat{v}' \vee \widehat{Null} \in \hat{v} \wedge \widehat{Null} \in \hat{v}' \wedge$
$\widehat{Undef} \notin \hat{v}' \vee \widehat{Undef} \in \hat{v} \wedge \widehat{Undef} \in \hat{v}'$

4.2 Abstract operations

4.3 Requirements verification

Chapter 5

Implementation

5.1 Constraint generation

Constraint elements: E.

$$\begin{array}{lll} \textit{Cache element} & \mathsf{C}(\ell) & : \ \mathcal{L} \to \hat{V} \\ \textit{Var element} & \Gamma(x) & : \ \mathcal{V} \to \hat{V} \\ \textit{State element} & \mathsf{M}(\mathcal{P}, ref) & : \ \mathcal{L} \times \mathcal{P} \to \hat{V} \end{array}$$

Permission Element: $\mathsf{P}(\ell) : \mathcal{L} \to \mathcal{P}$

Constraint form.

$$\begin{array}{lrcl} \textit{Term inclusion} & \{\hat{v}\} & \subseteq & \mathsf{E} \\ \textit{Element inclusion} & \mathsf{E} & \subseteq & \mathsf{E} \\ \textit{Permission inclusion} & \mathsf{P}(\ell) & \sqsubseteq & \mathsf{P}(\ell') \\ \textit{Operation} & \widehat{Op}(\overrightarrow{\mathsf{E}_i}) & \subseteq & \mathsf{E} \\ \textit{Implication} & \{\hat{v}\} \subseteq \mathsf{E} & \Rightarrow & \mathsf{E} \subseteq \mathsf{E} \end{array}$$

Misc:

r_*	is the set of all references of the program;
$lambda_*$	is the set of all lambdas of the program;

$$
\begin{array}{ll}
[\mathit{CG\text{-}Val}] & \mathcal{C}_{*\rho_s}[\![(c)^{\ell}]\!] = \{d_c\} \subseteq \mathsf{C}(\ell) \\
[\mathit{CG\text{-}Var}] & \mathcal{C}_{*\rho_s}[\![(x)^{\ell}]\!] = \mathsf{\Gamma}(x) \subseteq \mathsf{C}(\ell) \\
[\mathit{CG\text{-}Lambda}] & \mathcal{C}_{*\rho_s}[\![(\lambda x.{e_0}^{\ell_0})^{\ell}]\!] = \\
& \quad \{\{\lambda x.{e_0}^{\ell_0}\} \subseteq \mathsf{C}(\ell)\} \cup \\
& \quad \mathcal{C}_{*\rho_s}[\![({e_0}^{\ell_0})]\!] \\
[\mathit{CG\text{-}Obj}] & \mathcal{C}_{*\rho_s}[\![(\{\overrightarrow{str_i : {e_i}^{\ell_i}}\})^{\ell}]\!] = \\
& \quad \bigcup_i (\mathcal{C}_{*\rho_s}[\![({e_i}^{\ell_i})]\!] \cup \\
& \qquad \{\mathsf{P}(\ell_i) \sqsubseteq \mathsf{P}(\ell)\}) \cup \\
& \quad \{\{\overrightarrow{str_i : \mathsf{C}(\ell_i)}\} \subseteq \mathsf{C}(\ell)\} \\
[\mathit{CG\text{-}Let}] & \mathcal{C}_{*\rho_s}[\![(\mathbf{let}\ \overrightarrow{x_i = {e_i}^{\ell_i}}\ \mathbf{in}\ {e'}^{\ell'})^{\ell}]\!] = \\
& \quad \bigcup_i (\mathcal{C}_{*\rho_s}[\![({e_i}^{\ell_i})]\!] \cup \\
& \qquad \{\mathsf{C}(\ell_i) \subseteq \mathsf{\Gamma}(x_i)\} \cup \\
& \qquad \{\mathsf{P}(\ell_i) \subseteq \mathsf{P}(\ell)\}) \cup \\
& \quad \mathcal{C}_{*\rho_s}[\![({e'}^{\ell'})]\!] \cup \\
& \quad \{\mathsf{P}(\ell') \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\mathsf{C}(\ell') \subseteq \mathsf{C}(\ell)\} \\
[\mathit{CG\text{-}App}] & \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1}\, {e_2}^{\ell_2})^{\ell}]\!] = \\
& \quad \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1})]\!] \cup \mathcal{C}_{*\rho_s}[\![({e_2}^{\ell_2})]\!] \cup \\
& \quad \{\mathsf{P}(\ell_1) \sqsubseteq \mathsf{P}(\ell)\} \cup \{\mathsf{P}(\ell_2) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\{t\} \subseteq \mathsf{C}(\ell_1) \Rightarrow \mathsf{C}(\ell_2) \subseteq \mathsf{\Gamma}(x) \\
& \qquad | t = (\lambda x.{e_0}^{\ell_0}) \in lambda_*\} \cup \\
& \quad \{\{t\} \subseteq \mathsf{C}(\ell_1) \Rightarrow \mathsf{C}(\ell_0) \subseteq \mathsf{C}(\ell) \\
& \qquad | t = (\lambda x.{e_0}^{\ell_0}) \in lambda_*\} \cup \\
& \quad \{\{t\} \subseteq \mathsf{C}(\ell_1) \Rightarrow \mathsf{P}(\ell_0) \sqsubseteq \mathsf{P}(\ell) \\
& \qquad | t = (\lambda x.{e_0}^{\ell_0}) \in lambda_*\} \cup \\
[\mathit{CG\text{-}Op}] & \mathcal{C}_{*\rho_s}[\![(op(\overrightarrow{{e_i}^{\ell_i}}))^{\ell}]\!] = \\
& \quad \bigcup_i (\mathcal{C}_{*\rho_s}[\![({e_i}^{\ell_i})]\!] \cup \{\mathsf{P}(\ell_i) \sqsubseteq \mathsf{P}(\ell)\}) \cup \\
& \quad \{\widehat{op}(\mathsf{C}(\ell_i)) \subseteq \mathsf{C}(\ell)\} \\
[\mathit{CG\text{-}Cond}] & \mathcal{C}_{*\rho_s}[\![(\mathbf{if}\ ({e_0}^{\ell_0})\ \{\ {e_1}^{\ell_1}\ \}\ \mathbf{else}\ \{\ {e_2}^{\ell_2}\ \})^{\ell}]\!] = \\
& \quad \mathcal{C}_{*\rho_s}[\![({e_0}^{\ell_0})]\!] \cup \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1})]\!] \cup \mathcal{C}_{*\rho_s}[\![({e_2}^{\ell_2})]\!] \cup \\
& \quad \{\hat{P}(\ell_0) \sqsubseteq \hat{P}(\ell)\} \cup \\
& \quad \{\widehat{\mathbf{true}} \in \mathsf{C}(\ell_0) \Rightarrow \mathsf{C}(\ell_1) \subseteq \mathsf{C}(\ell)\} \cup \\
& \quad \{\widehat{\mathbf{true}} \in \mathsf{C}(\ell_0) \Rightarrow \mathsf{P}(\ell_1) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\widehat{\mathbf{false}} \in \mathsf{C}(\ell_0) \Rightarrow \mathsf{C}(\ell_2) \subseteq \mathsf{C}(\ell)\} \cup \\
& \quad \{\widehat{\mathbf{false}} \in \mathsf{C}(\ell_0) \Rightarrow \mathsf{P}(\ell_2) \sqsubseteq \mathsf{P}(\ell)\} \\
[\mathit{CG\text{-}While}] & \mathcal{C}_{*\rho_s}[\![(\mathbf{while}\ ({e_1}^{\ell_1})\ \{\ {e_2}^{\ell_2}\ \})^{\ell}]\!] = \\
& \quad \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1})]\!] \cup \mathcal{C}_{*\rho_s}[\![({e_2}^{\ell_2})]\!] \cup \\
& \quad \{\mathsf{P}(\ell_1) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\widehat{\mathbf{true}} \in \mathsf{C}(\ell_1) \Rightarrow \mathsf{C}(\ell_2) \subseteq \mathsf{C}(\ell)\} \cup \\
& \quad \{\widehat{\mathbf{true}} \in \mathsf{C}(\ell_1) \Rightarrow \mathsf{P}(\ell_2) \subseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\widehat{\mathbf{false}} \in \mathsf{C}(\ell_1) \Rightarrow \widehat{\mathbf{undefined}} \subseteq \mathsf{C}(\ell)\}
\end{array}
$$

$$
\begin{array}{ll}
[\mathit{CG\text{-}GetField}] & \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1}[{e_2}^{\ell_2}])^{\ell}]\!] = \\
& \quad \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1})]\!] \cup \mathcal{C}_{*\rho_s}[\![({e_2}^{\ell_2})]\!] \cup \\
& \quad \{\mathsf{P}(\ell_1) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\mathsf{P}(\ell_2) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \widehat{get}(\mathsf{C}(\ell_1), \mathsf{C}(\ell_2)) \subseteq \mathsf{C}(\ell) \\
[\mathit{CG\text{-}SetField}] & \mathcal{C}_{*\rho_s}[\![({e_0}^{\ell_0}[{e_1}^{\ell_1}] = {e_2}^{\ell_2})]\!] = \\
& \quad \mathcal{C}_{*\rho_s}[\![({e_0}^{\ell_0})]\!] \cup \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1})^{\ell}]\!] \cup \mathcal{C}_{*\rho_s}[\![({e_2}^{\ell_2})]\!] \cup \\
& \quad \{\mathsf{P}(\ell_1) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\mathsf{P}(\ell_2) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\mathsf{P}(\ell_3) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \widehat{set}(\mathsf{C}(\ell_1), \mathsf{C}(\ell_2), \mathsf{C}(\ell_2)) \subseteq \mathsf{C}(\ell) \\
[\mathit{CG\text{-}DelField}] & \mathcal{C}_{*\rho_s}[\![(\mathbf{delete}\ {e_1}^{\ell_1}[{e_2}^{\ell_2}])^{\ell}]\!] = \\
& \quad \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1})]\!] \cup \mathcal{C}_{*\rho_s}[\![({e_2}^{\ell_2})]\!] \cup \\
& \quad \{\mathsf{P}(\ell_1) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\mathsf{P}(\ell_2) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \widehat{del}(\mathsf{C}(\ell_1), \mathsf{C}(\ell_2)) \subseteq \mathsf{C}(\ell) \\
[\mathit{CG\text{-}Ref}] & \mathcal{C}_{*\rho_s}[\![(\mathbf{ref}_{r,\rho_r}\ {e_1}^{\ell_1})^{\ell}]\!] = \\
& \quad \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1})]\!] \cup \\
& \quad \{\{r\} \subseteq \mathsf{C}(\ell)\} \cup \\
& \quad \{\mathsf{P}(\ell_1) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\rho_r \sqsubseteq \rho_s \Rightarrow \mathsf{C}(\ell_1) \subseteq \mathsf{M}(r, \rho_r)\} \\
[\mathit{CG\text{-}DeRef}] & \mathcal{C}_{*\rho_s}[\![(\mathbf{deref}\ {e_1}^{\ell_1})^{\ell}]\!] = \\
& \quad \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1})]\!] \cup \\
& \quad \{\mathsf{P}(\ell_1) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{r \in \mathsf{C}(\ell_1) \Rightarrow \mathsf{M}(r, \rho_r) \subseteq \mathsf{C}(\ell) \\
& \quad\quad \mid r \in r_*, \rho_r \sqsubseteq \rho_s\} \\
[\mathit{CG\text{-}SetRef}] & \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1} = {e_2}^{\ell_2})^{\ell}]\!] = \\
& \quad \mathcal{C}_{*\rho_s}[\![({e_1}^{\ell_1})]\!] \cup \mathcal{C}_{*\rho_s}[\![({e_2}^{\ell_2})]\!] \cup \\
& \quad \{\mathsf{P}(\ell_1) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{\mathsf{P}(\ell_2) \sqsubseteq \mathsf{P}(\ell)\} \cup \\
& \quad \{r \in \mathsf{C}(\ell_1) \Rightarrow \mathsf{C}(\ell_2) \subseteq \mathsf{M}(r, \rho_r) \\
& \quad\quad \mid r \in r_*, \rho_r \sqsubseteq \rho_s\} \cup \\
& \quad \{\mathsf{C}(\ell_2) \subseteq \mathsf{C}(\ell)\} \\
[\mathit{PE\text{-}Send}] & \ldots \\
[\mathit{PE\text{-}Err}] & \ldots \\
[\mathit{PE\text{-}Exercise}] & \ldots
\end{array}
$$

5.2 Constraint solving

5.3 Implementation-specific details

Chapter 6

Experiments

6.1 Findings

6.2 Performance

SLOW... Very SLOW!!!

Chapter 7

Conclusion

7.1 Conclusions

7.2 Future works (unbundling)

[6]

References

[1] Chrome extension match pattern specification
https://developer.chrome.com/extensions/match_patterns, May 2014.

[2] Chrome extension overview
https://developer.chrome.com/extensions/overview, May 2014.

[3] Share me not extension
http://sharemenot.cs.washington.edu/, May 2014.

[4] Adam Barth, Adrienne Porter Felt, Prateek Saxena, and Aaron Boodman. Protecting browsers from extension vulnerabilities. Technical Report UCB/EECS-2009-185, EECS Department, University of California, Berkeley, Dec 2009.

[5] Nicholas Carlini, Adrienne Porter Felt, and David Wagner. An evaluation of the google chrome extension security architecture. In *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, pages 7–7, Berkeley, CA, USA, 2012. USENIX Association.

[6] Hanne Riis Nielson and Flemming Nielson. Flow logic: A multi-paradigmatic approach to static analysis. In *The Essence of Computation*, pages 223–244, 2002.