

DESCLASIFICACIÓN BASADA EN TIPOS EN DART

IMPLEMENTACIÓN Y ELABORACIÓN DE HERRAMIENTAS DE INFERENCIA

Matías Meneses Cortés

1. Control de flujo de información
2. Inferencia de tipos
3. Inferencia de facetas públicas en Dart
4. Elements
5. Conclusion

Control de flujo de información



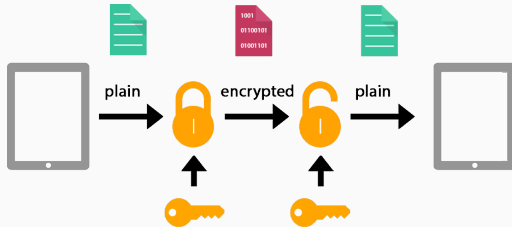
Protección de confidencialidad



Distintas técnicas de seguridad en distintas capas de comunicación.

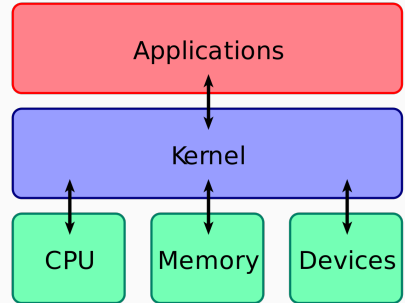
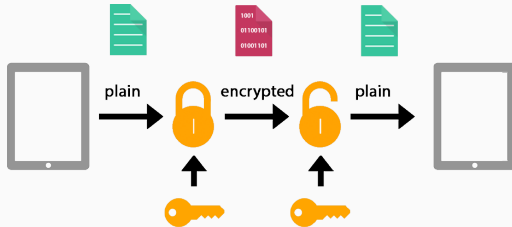
Protección de confidencialidad

Distintas técnicas de seguridad en distintas capas de comunicación.



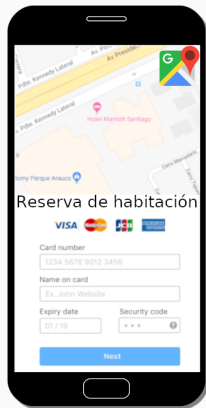
Protección de confidencialidad

Distintas técnicas de seguridad en distintas capas de comunicación.





Control de flujo de información



```
String book(String username, int date, int cardNumber) {  
    return sendToHotel(username, date, cardNumber);  
}
```

```
String sendToHotel(String username, int date, int cardNumber);  
String sendToGoogle(String token, int xCoord, int yCoord);
```

Control de flujo de información



```
String book(String username, int date, int cardNumber) {  
    return sendToGoogle (username, date, cardNumber);  
}
```

```
String sendToHotel(String username, int date, int cardNumber);  
String sendToGoogle(String token, int xCoord, int yCoord);
```

Tipado de seguridad para el control de flujo de información



```
String@L book(String@L username, int@L date, int@H cardNumber) {  
    return sendToGoogle(username, date, cardNumber);  
}
```

```
String@L sendToHotel(String@L username, int@L date, int@H cardNumber);  
String@L sendToGoogle(String@H token, int@L xCoord, int@L yCoord);
```

Tipado de seguridad para el control de flujo de información

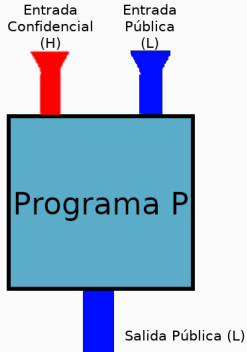


```
String@L book(String@L username, int@L date, int@H cardNumber) {  
    return sendToGoogle(username, date, cardNumber);  
}
```

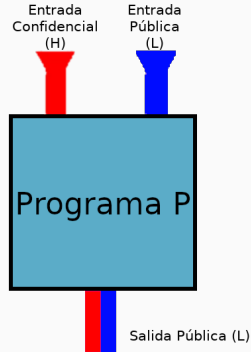
```
String@L sendToHotel(String@L username, int@L date, int@H cardNumber);  
String@L sendToGoogle(String@H token, int@L xCoord, int@L yCoord);
```

No-interferencia

Propiedad fundamental del control de flujo de información.



No-interferencia



Fuga de información

(asignación)

(asignación)

(argumento y parámetro)

Detección de flujos explícitos inválidos

(asignación)

(argumento y parámetro)

(retorno)

Detección de flujos implícitos inválidos

(codigo con if)

Detección de flujos implícitos inválidos

(codigo con if y marca de pc)

Contexto de seguridad (pc).

(Código de login)

(Código de login)

¡No cumple con no-interferencia!

(Código de login con declassify)

(Código de login con declassify(password))

(Código de login con declassify(password))

¡Grave fuga de información!

(código con tipos de dos facetas)

- `StringEq =`
 `[eq: String -> Bool]`
- `String <: StringEq`
 (Tipo bien formado)
- No-interferencia relajada

(figura de reticulo)

Reglas principales de la desclasificación basada en tipos

(código)

Métodos autorizados (TmD)

(regla formal)

Reglas principales de la desclasificación basada en tipos

(código)

Métodos no autorizados (TmH)

(regla formal)

- Propuesta sin implementación práctica.

- Propuesta sin implementación práctica.
- Anotación completa de facetas para realizar análisis.

- Propuesta sin implementación práctica.
- Anotación completa de facetas para realizar análisis.

(código con facetas públicas anotadas)

- Propuesta sin implementación práctica.
- Anotación completa de facetas para realizar análisis.

(código sin facetas públicas anotadas)

Inferencia de tipos

(codigo con anotaciones de tipo)

(codigo parcialmente anotado)

(codigo anotado con variables de tipo)

(codigo anotado con variables de tipo)

(restricciones generadas)

(Mostrar substituciones hasta resolver)

(codigo anotado con variables de tipo)
(retículo de subtipos)

(restricciones generadas)

Restricciones sobre subtipos

(codigo anotado con variables de tipo)
(retículo de subtipos mostrando meet y join)

(restricciones generadas)

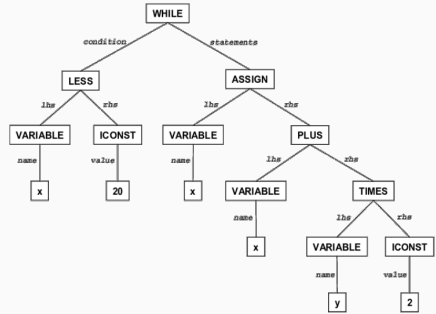
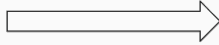
Objetivo de la memoria

Implementar un sistema de inferencia de facetas públicas para la desclasificación basada en tipos, en conjunto con una extensión para ambientes de desarrollo.

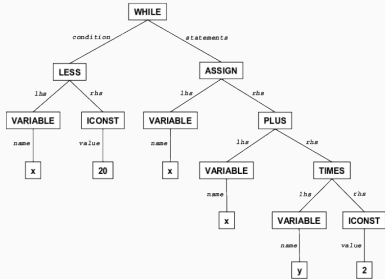
Inferencia de facetas públicas en Dart



Dart



Abstract Syntax Tree



Abstract Syntax Tree



Errores y sugerencias de edición


```
class Foo {
```

```
class Foo {  
  String foo(String a, String b) {
```



```
class Foo {  
  String foo(String a, String b) {  
    String s = "foo";  
  }  
}
```

```
class Foo {  
  String foo(String a, String b) {  
    String s = "foo";  
    if (a == b)
```

```
class Foo {  
  String foo(String a, String b) {  
    String s = "foo";  
    if (a == b) return a.concat(b);  
    return s;  
  }  
}
```

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código parcialmente tipado 1)

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código con tipos inferidos 1)

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código parcialmente tipado 2)

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código con tipos inferidos 2)

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código parcialmente tipado 3)

Problema de inferencia

Dado un programa Dart parcialmente tipado con facetas públicas, y completamente tipado con facetas privadas, encontrar la faceta pública de las expresiones no tipadas que más se ajuste al uso de las expresiones, tal que el programa sea bien tipado.

(código con tipos inferidos 3)

Uso de anotaciones de Dart para declarar las facetas públicas.

Uso de anotaciones de Dart para declarar las facetas públicas.
(código con `@S("StringEq")`)

Definición de facetas públicas

Uso de clases abstractas de Dart para declarar las facetas públicas.

Definición de facetas públicas

Uso de clases abstractas de Dart para declarar las facetas públicas.
(código con `abstract class StringEq`)

(código donde se usan tipos definidos por el usuario y tipos de dart)

(código donde se usan tipos definidos por el usuario y tipos de dart, destacando tipo definido por el usuario)

Conversión de tipos de Dart a facetas privadas

(código donde se usan tipos definidos por el usuario y tipos de dart, destacando tipo estático común de dart)

(Mostrar operación de convert)

(Mostrar operación de convert)

P_{Bi} (algo)

(Mostrar operación de convert)

P_{Ai} (algo)

The **METROPOLIS** theme is a Beamer theme with minimal visual noise inspired by the HSRM Beamer Theme by Benjamin Weiss.

Enable the theme by loading

```
\documentclass{beamer}  
\usetheme{metropolis}
```

Note, that you have to have Mozilla's *Fira Sans* font and XeTeX installed to enjoy this wonderful typography.

Sections group slides of the same topic

```
\section{Elements}
```

for which **METROPOLIS** provides a nice progress indicator ...

METROPOLIS supports 4 different titleformats:

- Regular
- SMALLCAPS
- ALLSMALLCAPS
- ALLCAPS

They can either be set at once for every title type or individually.

This frame uses the `smallcaps` titleformat.

Potential Problems

Be aware, that not every font supports small caps. If for example you typeset your presentation with pdfTeX and the Computer Modern Sans Serif font, every text in smallcaps will be typeset with the Computer Modern Serif font instead.

This frame uses the `allsmallcaps` titleformat.

Potential problems

As this titleformat also uses smallcaps you face the same problems as with the `smallcaps` titleformat. Additionally this format can cause some other problems. Please refer to the documentation if you consider using it.

As a rule of thumb: Just use it for plaintext-only titles.

This frame uses the `allcaps` titleformat.

Potential Problems

This titleformat is not as problematic as the `allsmallcaps` format, but basically suffers from the same deficiencies. So please have a look at the documentation if you want to use it.

Elements

The theme provides sensible defaults to
`\emph{emphasize}` text, `\alert{accent}` parts
or show `\textbf{bold}` results.

becomes

The theme provides sensible defaults to *emphasize* text, **accent** parts or show **bold** results.

Font feature test

- Regular
- *Italic*
- SMALLCAPS
- Bold
- *Bold Italic*
- BOLD SMALLCAPS
- Monospace
- Monospace Italic
- Monospace Bold
- Monospace Bold Italic

Items

- Milk
- Eggs
- Potatos

Enumerations

1. First,
2. Second and
3. Last.

Descriptions

PowerPoint Meeh.

Beamer Yeeeha.

- This is important

- This is important
- Now this

- This is important
- Now this
- And now this

- This is really important
- Now this
- And now this

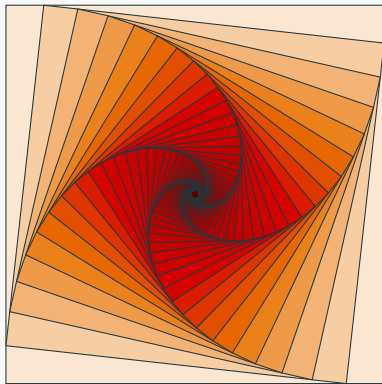


Figure 1: Rotated square from texample.net.

Table 1: Largest cities in the world (source: Wikipedia)

City	Population
Mexico City	20,116,842
Shanghai	19,210,000
Peking	15,796,450
Istanbul	14,160,467

Blocks

Three different block environments are pre-defined and may be styled with an optional background color.

Default

Block content.

Alert

Block content.

Example

Block content.

Default

Block content.

Alert

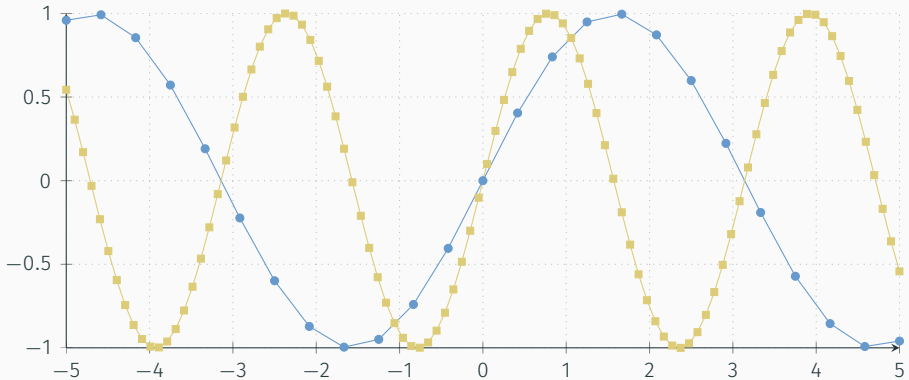
Block content.

Example

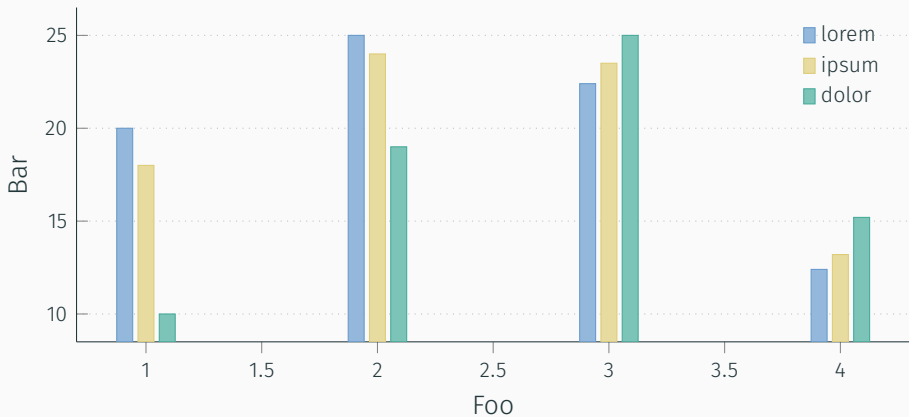
Block content.

$$e = \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n$$

Line plots



Bar charts



Veni, Vidi, Vici

METROPOLIS defines a custom beamer template to add a text to the footer. It can be set via

```
\setbeamertemplate{frame footer}{My custom footer}
```

Some references to showcase `[allowframebreaks]` [4, 2, 5, 1, 3]

Conclusion

Get the source of this theme and the demo presentation from

`github.com/matze/mtheme`

The theme *itself* is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.






Questions?

Backup slides

Sometimes, it is useful to add slides at the end of your presentation to refer to during audience questions.

The best way to do this is to include the **appendixnumberbeamer** package in your preamble and call `\appendix` before your backup slides.

METROPOLIS will automatically turn off slide numbering and progress bars for slides in the appendix.

-  P. Erdős.
A selection of problems and results in combinatorics.
In *Recent trends in combinatorics* (Matrahaza, 1995), pages 1–6. Cambridge Univ. Press, Cambridge, 1995.
-  R. Graham, D. Knuth, and O. Patashnik.
Concrete mathematics.
Addison-Wesley, Reading, MA, 1989.
-  G. D. Greenwade.
The Comprehensive Tex Archive Network (CTAN).
TUGBoat, 14(3):342–351, 1993.



D. Knuth.

Two notes on notation.

Amer. Math. Monthly, 99:403–422, 1992.



H. Simpson.

Proof of the Riemann Hypothesis.

preprint (2003), available at <http://www.math.drofnats.edu/riemann.ps>,
2003.