

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. Validación
5. Conclusión

Control de flujo de información



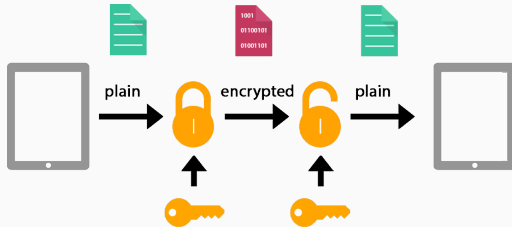
Protección de confidencialidad



Distintas técnicas de seguridad en distintas capas de comunicación.

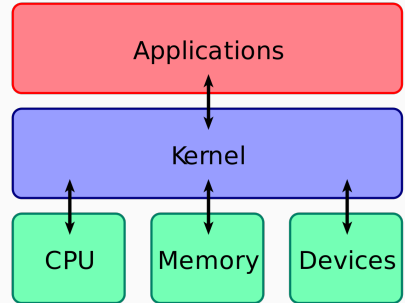
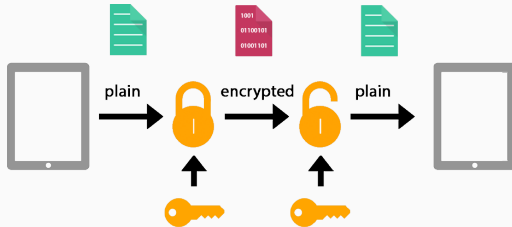
Protección de confidencialidad

Distintas técnicas de seguridad en distintas capas de comunicación.



Protección de confidencialidad

Distintas técnicas de seguridad en distintas capas de comunicación.



Seguridad basada en el lenguaje: Tipado de seguridad



```
String@L login(String@L guess, String@H password) {  
    if (password == guess) return "Login successful";  
    else return "Login failed";  
}
```

H
|
L

Orden parcial de dos niveles

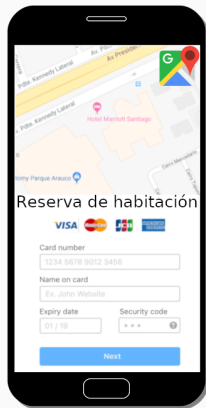
Control de flujo de información



```
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);
```

Tipado de seguridad para el control de flujo de información

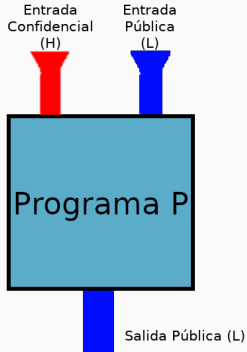


```
String@L book(String@L username, int@L date, int@H cardNumber) {  
    return sendToHotel(username, date, cardNumber);  
}
```

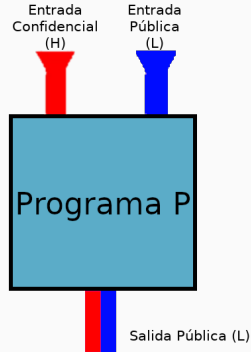
```
String@L sendToHotel(String@L username, int@L date, int@H cardNumber);  
String@L sendToGoogle(String@H token, int@L xCoord, int@L yCoord);
```

No-interferencia

Propiedad fundamental del control de flujo de información.



No-interferencia



Fuga de información

Detección de flujos explícitos inválidos

```
String@H password = "abc";  
String@L s = password;
```


Detección de flujos explícitos inválidos

```
tellEveryone(String@L s) {  
    shout(s);  
}
```

```
String@H secret = "Me dan miedo las polillas";  
tellEveryone(secret);
```

Detección de flujos explícitos inválidos

```
String@L tellSecret(String@H secret) {  
    return secret;  
}
```

Detección de flujos implícitos inválidos

```
String@L login(String@L guess, String@H password) {  
    if (password == guess) return "Login successful";  
    else return "Login failed";  
}
```

Detección de flujos implícitos inválidos

```
String@L login(String@L guess, String@H password) {  
  if (password == guess) return "Login successful";  
  else return "Login failed";  
}
```

Contexto de seguridad (pc).

```
String@L login(String@L guess, String@H password) {  
    if (password == guess) return "Login successful";  
    else return "Login failed";  
}
```

¡No cumple con no-interferencia!

```
String@L login(String@L guess, String@H password) {  
    if (declassify(password == guess)) return "Login successful";  
    else return "Login failed";  
}
```

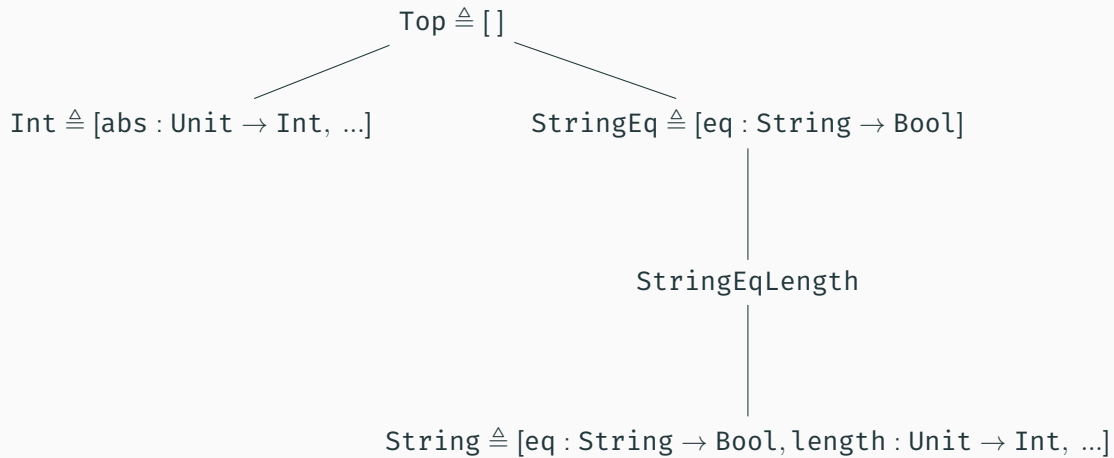
```
declassify(password)
```

```
declassify(password)
```

¡Grave fuga de información!


```
String<String login(String<String guess, String<StringEq password) {  
  if (password.eq(guess)) return "Login successful";  
  else return "Login failed";  
}
```

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



Regla principal de la desclasificación basada en tipos

```
String<String login(String<String guess, String<Top password) {  
    if ( password.eq(guess) ) return "Login successful";  
    else return "Login failed";  
}
```

Regla principal de la desclasificación basada en tipos

```
String<String login(String<String guess, String<Top password) {  
    if ( password.eq(guess) ) return "Login successful";  
    else return "Login failed";  
}
```

- 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)

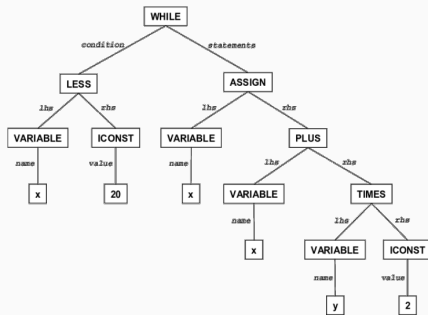
Objetivo de la memoria

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

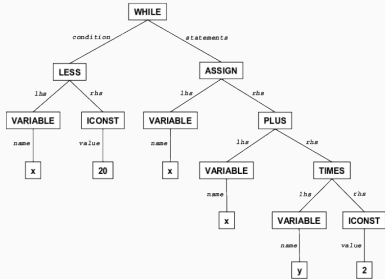
Inferencia de facetas públicas en Dart



Dart



Abstract Syntax Tree



Abstract Syntax Tree



Errores y sugerencias de edición

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)

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

```
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;  
  }  
}
```

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

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`)

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Conversión de tipos de Dart a facetas privadas

(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)

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Gramática de tipos

$$\tau := \alpha \mid \text{Obj}(\overline{l : \tau}) \mid \overline{\tau} \rightarrow \tau \mid \tau \sqcap \tau \mid \tau \sqcup \tau \mid \text{Bot} \mid \text{Top}$$

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

(Código con invocación a método y variables de tipo) (restricciones generadas)

(Código con expresión de retorno y variables de tipo) (restricciones generadas)

(Código con expresión de asignación y variables de tipo) (restricciones generadas)

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Simplificación de restricciones

Simplificación de restricciones
(Restricciones no simplificadas)

Simplificación de restricciones
(Restricciones no simplificadas con obvias marcadas)

Simplificación de restricciones
(Restricciones simplificadas)

Agrupación de restricciones
(Restricciones no agrupadas)

Agrupación de restricciones

(Marcar con distinto color las restricciones de un grupo u otro)

Agrupación de restricciones
(Restricciones agrupadas)

Unificación: Construcción de tipos
(Restricciones agrupadas)

Unificación: Construcción de tipos
(Restricciones agrupadas con candidatos a construcción marcados)

Unificación: Construcción de tipos
(Restricciones con nuevos tipos)

Unificación: Verificación de restricciones
(Restricciones)

Unificación: Verificación de restricciones

(Restricciones con relaciones inválidas que provienen de invocación a método marcadas)

Unificación: Verificación de restricciones
(Restricciones actualizadas luego de comprobación)

Unificación: Verificación de restricciones

(Restricciones con relaciones inválidas que no provienen de invocación a método marcadas)

Unificación: Verificación de restricciones
(Error generado por la restricción no válida)

Unificación: Substitución de restricciones resueltas
(Restricciones)

Unificación: Substitución de restricciones resueltas
(Restricciones con restricciones resueltas marcadas)

Unificación: Substitución de restricciones resueltas
(Substitución de restricciones)

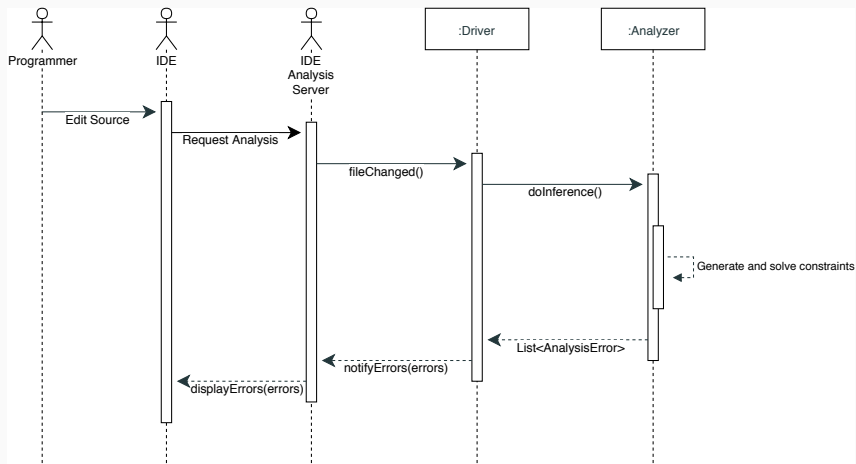
Unificación: Algoritmo iterativo

Algoritmo de unificación

(pseudocódigo del algoritmo)

1. Definición del subconjunto de Dart soportado.
2. Definición y declaración de facetas públicas en Dart.
3. Integración de tipos estáticos comunes de Dart con las facetas de la desclasificación basada en tipos.
4. Definición de la gramática de tipos.
5. Descripción del paso de generación de restricciones para un programa Dart.
6. Descripción del paso de resolución de restricciones.
7. Implementación de la extensión para ambientes de desarrollo.

Extensión para ambientes de desarrollo



`SecurityError`

`SecurityError`

(código que genera una restricción inválida de esas)

`IllFormedTypeError`

IllFormedTypeError

(código que posee un tipo mal formado)

UndefinedFacetWarning

`UndefinedFacetWarning`

(Código con faceta pública no definida)

InferredFacetInfo

InferredFacetInfo

(código con un tipo inferido)

Validación

Ejemplo: Sistema de autenticación web

Secure Login

Enter your credentials

Enter your email

Enter your password

[Forgot Password?](#)

LOGIN

Ejemplo: Sistema de autenticación web

(imagen en intelliJ definiendo la base de datos)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ definiendo el metodo login)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ mostrando la inferencia)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ declarando una faceta pública no definida)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ con el warning sobre la faceta no definida)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ usando libreria html)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ agregando Top a variable de contraseña)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ del error de seguridad)

Ejemplo: Sistema de autenticación web

(imagen en intelliJ corrigiendo el error)

Ejemplo: Sistema de autenticación web

Secure Login

Enter your credentials

Enter your email

Enter your password

[Forgot Password?](#)

LOGIN

Tabla 1: Anotación de facetas públicas en identificadores

Con inferencia	Sin inferencia
7	22

Test unitarios en el repositorio del proyecto.

Conclusión

Conexión entre abstracciones de tipo y relaciones de orden de etiquetas de seguridad.

Formalización de inferencia.

Extensión al subconjunto soportado de Dart.

Características de la extensión para ambientes de desarrollo.

Extensión a polimorfismo.

Extensión a polimorfismo.
(lista parametrizada)

Preguntas

Cosas extra por posibles preguntas