GADTs meet their Match

Tom Schrijvers

with Georgios Karachalas, Dimitrios Vytiniotis and Simon Peyton Jones
Checking Pattern Matches
Property 1: Exhaustiveness

```
zip []     []     = []
zip (a:as) (b:bs) = (a,b) : zip as bs
```
Property I: Exhaustiveness

\[
\text{zip } [] \quad [] \quad = \quad [] \\
\text{zip } (a:as) \quad (b:bs) \quad = \quad (a,b) : \quad \text{zip as bs}
\]

*Main> zip [] [1]  
*** Exception:  
Non-exhaustive patterns in function zip
Property 1: Exhaustiveness

```
zip [] [] = []
zip (a:as) (b:bs) = (a,b) : zip as bs
```

*Main> zip [] [1]
*** Exception: Non-exhaustive patterns in function zip

Runtime crash 😞
Property 1: Exhaustiveness

\[
\begin{align*}
\text{zip} \; [] \; [] & = [] \\
\text{zip} \; (a:\text{as}) \; (b:\text{bs}) & = (a,b) : \text{zip} \; \text{as} \; \text{bs}
\end{align*}
\]
Property 1: Exhaustiveness

\[
\begin{align*}
\text{zip} \; [] & \; [] = [] \\
\text{zip} \; (a:a\text{!}) \; (b:b\text{!}) & = (a,b) : \text{zip} \; a\text{!} \; b\text{!}
\end{align*}
\]

*Main*> :l zip.hs
[1 of 1] Compiling Main

Warning:

Pattern match(es) are non-exhaustive
In an equation for ‘zip’:
Patterns not matched:

\[
[] \; (_ : _) \\
(_ : _) \; []
\]
Property 1: Exhaustiveness

\[
\text{zip} \ [\] \ [\] = [] \\
\text{zip} \ (a:as) \ (b:bs) = (a,b) : \text{zip} \ as \ bs
\]

*Main> :l zip.hs
[1 of 1] Compiling Main
Warning: Pattern match(es) are non-exhaustive.
In an equation for ‘zip’:
  Patterns not matched:
    [] \(_ : \_\)
    (\_ : \_ \) []

Compiler warning 😃
GADT Problem 1

data Vect :: Nat -> * -> * where
  VN :: Vect Zero a
  VC :: a -> Vect n a -> Vect (Succ n) a

vzip :: Vect n a -> Vect n b -> Vect n (a,b)
vzip VN VN = VN
vzip (VC x xs) (VC y ys) = VC (x,y) (vzip xs ys)
GADT Problem I

data Vect :: Nat -> * -> * where
  VN :: Vect Zero a
  VC :: a -> Vect n a -> Vect (Succ n) a

vzip :: Vect n a -> Vect n b -> Vect n (a,b)
vzip VN VN = VN
vzip (VC x xs) (VC y ys) = VC (x,y) (vzip xs ys)

Pattern match(es) are non-exhaustive
In an equation for ‘vzip’:
Patterns not matched:
  VN (VC _ _)
  (VC _ _)VN
GADT Problem 1

data Vect :: Nat -> * -> * where
  VN :: Vect Zero a
  VC :: a -> Vect n a -> Vect (Succ n) a

vzip :: Vect n a -> Vect n b -> Vect n (a,b)
vzip VN VN = VN
vzip (VC x xs) (VC y ys) = VC (x,y) (vzip xs ys)

Pattern match(es) are non-exhaustive
In an equation for 'vzip':
Patterns not matched:
  VN (VC _ _) (VC _ _ ) VN

bogus warning!
Property 2: Redundancy

\[
\begin{align*}
\text{len } [] & = 0 \\
\text{len } (x:xs) & = 1 + \text{len } xs \\
\text{len } l & = 42
\end{align*}
\]
Property 2: Redundancy

len [] = 0
len (x:xs) = 1 + len xs
len l = 42
Property 2:

Redundancy

\[
\begin{align*}
\text{len} \; [\;] &= 0 \\
\text{len} \; (x:xs) &= 1 + \text{len} \; xs \\
\text{len} \; l &= 42
\end{align*}
\]

Warning:
Pattern match(es) are overlapped
In an equation for ‘len’: len l = ...

dead code
Property 2: Redundancy

\[
\begin{align*}
\text{len } [] &= 0 \\
\text{len } (x:xs) &= 1 + \text{len } xs \\
\text{len } l &= 42
\end{align*}
\]

Warning:
Pattern match(es) are overlapped
In an equation for ‘len’: len l = ...
GADT Problem 2

\[ \text{vzip } VN \text{ VN } = VN \]
\[ \text{vzip } (\text{VC } x \text{ xs}) \text{ (VC y ys)} = \text{VC } (x,y) \text{ (vzip xs ys)} \]
\[ \text{vzip } _ {\text{ }}_{\text{ }}_{\text{ }}_{\text{ }} = \text{error "unreachable" } \]
GADT Problem 2

\[
\begin{align*}
\text{vzip } \text{VN } \text{VN} & = \text{VN} \\
\text{vzip } (\text{VC } x \ x \ x) (\text{VC } y \ y \ y) & = \text{VC } (x,y) (\text{vzip } x \ x \ y \ y) \\
\text{vzip } _ & = \text{error } "\text{unreachable}" \\
\end{align*}
\]

suppresses bogus non-exhaustiveness warnings
GADT Problem 2

vzip VN VN = VN
vzip (VC x xs) (VC y ys) = VC (x,y) (vzip xs ys)
vzip _ _ = error “unreachable”

suppresses bogus non-exhaustiveness warnings

dead code
GADT Problem 2

\[
\begin{align*}
\text{vzip VN VN} &= VN \\ \\
\text{vzip (VC x xs) (VC y ys)} &= VC (x,y) (\text{vzip xs ys}) \\ \\
\text{vzip _ ___} &= \text{error “unreachable”}
\end{align*}
\]

- suppresses bogus non-exhaustiveness warnings
- dead code
- No compiler warning

😢
💀

7
Quick Summary

Exhaustiveness
Redundancy

\{\}

broken for GADTs
Quick Summary

Exhaustiveness

Redundancy

But there is more ...

broken for GADTs
Challenge 1

data F a where
  F1 :: F Int
  F2 :: F Bool

data G a where
  G1 :: G Int
  G2 :: G Char

h :: F a -> G a -> Int
h F1 G1 = 1
h _ _ = 2
Challenge 1

data F a where
F1 :: F Int
F2 :: F Bool

data G a where
G1 :: G Int
G2 :: G Char

h :: F a -> G a -> Int
h F1 G1 = 1
h _ _ = 2

redundant?
Challenge 1

data F a where
  F1 :: F Int
  F2 :: F Bool

data G a where
  G1 :: G Int
  G2 :: G Char

h :: F a -> G a -> Int

h F1 G1 = 1
h _ _ = 2

> h F2 undefined
2

redundant?
Challenge 1

\[
data F \ a \ where \\
   F1 :: F \ Int \\
   F2 :: F \ Bool \\
\]

\[
data G \ a \ where \\
   G1 :: G \ Int \\
   G2 :: G \ Char \\
\]

\[
h :: F \ a \rightarrow G \ a \rightarrow Int \\
h \ F1 \ G1 = 1 \\
\underline{h \ _ \ _ = 2} \\
\]

redundant?

> h F2 undefined 2
Challenge 1

data F a where
  F1 :: F Int
  F2 :: F Bool

h :: F a -> G a -> Int
h F1 G1 = 1
h _ _ = 2

> h F2 undefined
2
> h F2 undefined
*** Exception

data G a where
  G1 :: G Int
  G2 :: G Char
Challenge 2

g :: Bool -> Bool -> Int
g _ False = 1
g True False = 2
g _ _ = 3
Challenge 2

g :: Bool -> Bool -> Int

- False = 1
- True False = 2
- _ _ = 3
Challenge 2

```
g :: Bool -> Bool -> Int

g False False = 1

g True False = 2

> g undefined True

*** Exception
```
Challenge 2

g :: Bool -> Bool -> Int
g False _ = 1
_ True False = 2
_ _ _ = 3

> g undefined True
*** Exception

> g undefined True
3
Challenge 2

\[
\begin{align*}
g & : \text{Bool} \to \text{Bool} \to \text{Int} \\
g \_ \_ \text{False} & = 1 \\
g \text{True} \_ \text{False} & = 2 \\
g \_ \_ \_ & = 3
\end{align*}
\]

\[\text{> g undefined True} \quad \text{*** Exception}\]

\[\text{> g undefined True} \quad 3\]
Challenge 2

\texttt{\textbackslash\textbackslash use\textbackslash\textbackslash useful\textbackslash\textbackslash pattern}

\begin{verbatim}
g :: Bool -> Bool -> Int
g _ False = 1
g True False = 2
g _ _ = 3
\end{verbatim}

\texttt{\textbackslash\textbackslash unreachable\textbackslash\textbackslash RHS}

\texttt{> g undefined True
*** Exception}

\texttt{> g undefined True
3}
The Guard Challenge

abs2 :: Int -> Int
abs2 x  | x < 0  = -x
         | x >= 0 = x

exhaustive?
The Guard Challenge

```
abs2 :: Int -> Int
abs2 x | x < 0  = -x
       | x >= 0 = x

append xs ys
  | []    <- xs = ys
  | (p:ps) <- xs = p : append ps ys
```
What we need:

A uniform approach to deal with:

- GADTs
- Laziness
- Guards
Our Approach
Symbolic representation of all possible values

\[ U_0 \]

\[ p_{11}\ldots p_{1n} \]

\[ \text{patVectProc} \]

\[ U_1 \]

\[ p_{21}\ldots p_{2n} \]

\[ \text{patVectProc} \]

[...]

\[ p_{m1}\ldots p_{mn} \]

\[ \text{patVectProc} \]

Symbolic representation of all uncovered values

\[ U_n \]
Basic ADT Support
zip [] [] = ...
zip (x:xs) (y:ys) = ...
zip [ ] [ ] = ...
zip (x:xs) (y:ys) = ...
\([\ ]\)[\[]

\((x:xs)\ (y:ys)\)
(x:xs) (y:ys)
all possible values

\[(x:xs) \ (y:ys)\]
(x:xs) (y:ys)
all possible values

\[
\begin{array}{c}
\{ \_ \_ \} \\
\end{array}
\]

\[
\begin{array}{c}
\text{[ ] [ ]} \\
\end{array}
\]

\[
\begin{array}{c}
\text{[ [ ] [ ]]} \\
\end{array}
\]

\[
\begin{array}{c}
D_1 \\
\end{array}
\]

\[
\begin{array}{c}
U_1 \\
\end{array}
\]

\[
(x:xs) \ (y:ys)
\]
(x:xs) (y:ys)
\[(x:xs) (y:ys)\]
(x:xs) (y:ys)
all possible values

useful clause

forcing

\[(x:xs) (y:ys)\]
all possible values

useful clause

forcing

{x:xs} {y:ys}

C_2

D_2

U_2

{(\_\_\_ \_), [ ] (\_\_\_)}

{ [ ] [ ] }

{\perp \_, [ ] \perp}

{ [ ] [ ]}
all possible values

[ ] [ ]

{( _ : _ ) _, [ ] ( _ : _ )}

(x : xs) (y : ys)

{( _ : _ ) ( _ : _ )}

{ _ _ }

D₂

U₂

useful clause

forcing
all possible values

\{\_\_\_\}\n
useful clause

\{\ [\ ]\ [\ ]\}\n
forcing

\{\_\_\, [\ ] \perp\}\n
(x:xs) (y:ys)

\{(\_\_\_\) \_\, [\] \(_\_\_\))\n
U_2

\{(\_\_\_\) \(_\_\_\)\n
\{(\_\_\_\) \_\}\
all possible values

\[
\{ \_ \_ \} \]

useful clause

\[
\{ \_ \_ , [ ] \_ \}
\]

forcing

\[
\{(_:_)(_:_)\}
\]

\[
\{(_:_)[],[],(_:_)(_:_)\}
\]

\[
(x:xs) (y:ys)
\]

\[
\{(_:_)(_:_)\}
\]

\[
\{(_:_)[],[],(_:_)(_:_)\}
\]
all possible values

{ _ _ }

useful clause

{ [ ] [ ] }

forcing

{ _ _ , [ ] _ }

not exhaustive

{( _ _ ) _ , [ ] ( _ _ )}

(x:xs) (y:ys)

{( _ _ ) ( _ _ )}

{( _ _ ) _ }

{( _ _ ) [ ] , [ ] ( _ _ )}
Basic Syntax

**Pattern**

\[ p ::= _ \mid K \bar{p} \]

**Value Abstraction**

\[ u ::= _ \mid K \bar{u} \]
Clause Processing: Covered Values

\[ C \hat{p} \hat{u} = c \]
Clause Processing:
Covered Values

\[ C \vec{p} \vec{u} = C \]

\[ C \varepsilon \varepsilon = \{\varepsilon\} \]
Clause Processing: Covered Values

\[ C \, \vec{p} \, \vec{u} = C \]

\[ C \, \varepsilon \, \varepsilon = \{ \varepsilon \} \]

\[ C \, (\_ \, \vec{p}) \, (u \, \vec{u}) = \{ u \, \vec{w} \mid \vec{w} \leftarrow C \, \vec{p} \, \vec{u} \} \]
Clause Processing: Covered Values

\[ C \hat{p} \hat{u} = C \]

\[ C \varepsilon \varepsilon = \{ \varepsilon \} \]

\[ C (\_ \hat{p}) (u \hat{u}) = \{ u \hat{w} | \hat{w} <- C \hat{p} \hat{u} \} \]

\[ C ((K_i \hat{q}) \hat{p}) ((K_j \hat{w}) \hat{u}) \]
\[ | K_i == K_j = \{ (K_i \hat{w}') \hat{u}' \}
   | \hat{w}' \hat{u}' <- C (\hat{q} \hat{p}) (\hat{w} \hat{u}) \}
\[ | \text{otherwise} = \{ \} \]
Clause Processing:
Covered Values

\[ C \overrightarrow{p}\overrightarrow{u} = C \]

\[ C \ \varepsilon \ \varepsilon = \{ \varepsilon \} \]

\[ C \ (_\overrightarrow{p}) \ (u \ \overrightarrow{u}) = \{ u \ \overrightarrow{w} \mid \overrightarrow{w} \leftarrow C \overrightarrow{p}\overrightarrow{u} \} \]

\[ C \ ((K_i \ \overrightarrow{q}) \ \overrightarrow{p}) \ ((K_j \ \overrightarrow{w}) \ \overrightarrow{u}) \]

\[ \mid K_i = K_j = \{(K_i \ \overrightarrow{w}') \ \overrightarrow{u}' \mid \overrightarrow{w}' \ \overrightarrow{u}' \leftarrow C \ (\overrightarrow{q} \overrightarrow{p}) \ (\overrightarrow{w} \ \overrightarrow{u}) \} \]

\[ \mid \text{otherwise} = \{ \} \]

\[ C \ ((K_i \ \overrightarrow{q}) \ \overrightarrow{p}) \ (_\overrightarrow{u}) \]

\[ = C \ ((K_i \ \overrightarrow{q}) \ \overrightarrow{p}) \ ((K_i \ \_\_\_) \ \overrightarrow{u}) \]
Clause Processing: Covered Values

\( C \vec{p} \vec{u} = C \)

Uncovered
\( \vec{u} \vec{p} \vec{u} = U \)

Diverging
\( \vec{D} \vec{p} \vec{u} = D \)

\( C \ ( (K_i \ vec{q}) \vec{p}) \ _\vec{u} \)
\( = C \ ( (K_i \ vec{q}) \vec{p}) \ ((K_i \ _\vec{u}) \vec{u}) \)

\( = \{ \varepsilon \} \)

\( = \{ u \ vec{w} \mid \ vec{w} \leftarrow C \vec{p} \vec{u} \} \)

\( (K_j \ vec{w}) \vec{u} \)
\( \{(K_i \ vec{w'}) \vec{u}' \mid vec{w'} \vec{u}' \leftarrow C (\vec{q} \vec{p}) (\vec{w} \vec{u}) \} \)
\( = \{ \} \)

\( \vec{u} \vec{p} \vec{u} = U \)

\( \vec{D} \vec{p} \vec{u} = D \)
GADTs
GADTs and Type Constraints

data Vec :: Nat -> * -> * where
  VN :: Vec Z a
  VC :: a -> Vec m a -> Vec (S m) a
GADTs and Type Constraints

data Vec :: Nat -> * -> * where
  VN :: n ~ Z       => Vec Z a
  VC :: n ~ S m    => a -> Vec m a -> Vec (S m) a

Equivalent

data Vec :: Nat -> * -> * where
  VN :: n ~ Z       => Vec n a
  VC :: n ~ S m    => a -> Vec m a -> Vec n a
GADTs and Type Constraints

data Vec :: Nat -> * -> * where
    VN ::                 Vec Z a
    VC :: a -> Vec m a -> Vec (S m) a

equivalent

data Vec :: Nat -> * -> * where
    VN :: n ~ Z =>                 Vec n a
    VC :: n ~ S m => a -> Vec m a -> Vec n a

type constraints
Syntax with Type Constraints

Value Abstraction

\[ v ::= \Gamma \vdash \tilde{u} \triangleright \Delta \]

\[ u ::= _ | K \tilde{u} \]

\[ \Gamma ::= \varepsilon | \Gamma, a \]

\[ \Delta ::= \varepsilon | \Delta \cup \Delta | \tau \sim \tau \]
The Oracle
The Oracle

Δ

Not Satisfiable
\{ \_ \_ \_ \rightarrow \varepsilon \}
\{\_ \_ \_ \varepsilon\}

unconstrained
unconstrained

\{ \_ \_ \n跑 \epsilon \}

\{ VN \ VN
\n跑 \ (n-Z) \cup (n-Z) \}
unconstrained

\{ \_ \_ \{ VN VN \{ ε \}

\{ VN VN \{ (n\sim Z) \cup (n\sim Z) \}

\{ \perp \_ \{ ε \}

\{ VN \perp \{ n\sim Z \}

VN VN
\{ __ __ \triangleright \varepsilon \}\n
\{ VN VN \triangleright (n \sim Z) U (n \sim Z) \}\n
\{ \bot __ \triangleright \varepsilon , VN \bot \triangleright n \sim Z \}\n
\{(VC __ __) __ \triangleright n \sim S m , VN (VC __ __) \triangleright (n \sim Z) U (n \sim S m)\}\n
unconstrained
\[
\{\_\_ \triangleright \varepsilon\}
\]

\[
\Rightarrow \{ VN \ VN \\
\triangleright (n \sim Z) \cup (n \sim Z) \}
\]

\[
\{ \bot \_ \triangleright \varepsilon \\
, VN \bot \triangleright n \sim Z \}
\]

\[
\{(VC \_\_) \_ \triangleright n \sim S \ m \\
, VN (VC \_\_) \triangleright (n \sim Z) \cup (n \sim S \ m) \}
\]

unconstrained

inconsistent
\[
\{ \_ \_ \ \triangleright \ \varepsilon \}
\]

\[
\{ \text{VN VN} \ \triangleright \ (n \sim Z) \cup (n \sim Z) \}
\]

\[
\{ \bot \_ \ \triangleright \ \varepsilon \\
, \text{VN} \ \bot \ \triangleright \ n \sim Z \}
\]

\[
\{(VC \ _ \ _) \ _ \ \triangleright \ n \sim S \ m \\
, \text{VN} \ (VC \ _ \ _) \ \triangleright \ (n \sim Z) \cup (n \sim S \ m) \}
\]

unconstrained

inconsistent
continued

\[
\{ (VC \_\_\_) \_ \triangleright n \sim S \; m \}
\]

(VC \; x \; xs) \; (VC \; y \; ys)
continued

\{ (VC _ _) _ \triangleright n \sim S m \}\n
(VC x xs) (VC y ys)

\{ (VC _ _) (VC _ _) \triangleright (n \sim S m) U (n \sim S o) \}\n
\[\{ (\text{VC } _ _ ) _ _ \uparrow \ n \sim S \ m \}\]

\((\text{VC } x \; x s) \; (\text{VC } y \; y s)\)

\[\{ (\text{VC } _ _ ) \; (\text{VC } _ _ ) \uparrow (n\sim S \; m) \cup (n\sim S \; o)\}\]

\[\{ (\text{VC } _ _ ) \Downarrow \uparrow n\sim S \; m\}\]
continued

\{ (VC \_\_\_) \_ \n \sim S \ m \}\n
(VC \ x \ xs) (VC \ y \ ys) \n\{ (VC \_\_\_) (VC \_\_\_) \n \sim S \ m \} \cup \{ (VC \_\_\_) \n \sim S \ o \}\n
\n\{ (VC \_\_\_) \n \sim S \ m \}\n
\{ (VC \_\_\_) VN \n \sim S \ m \} \cup \{ (n \sim S \ m) \cup (n \sim Z) \}\n
continued

\{(VC \_\_\_) \_ \n \sim S m\}

(VC x xs) (VC y ys) \n\rightarrow \n\{ (VC \_\_\_) (VC \_\_\_) \n \n \sim S m) \cup (n \sim S o) \n\}

\{(VC \_\_\_) VN \n \n \sim S m\}

\{(VC \_\_\_) VN \n \n \sim S m) \cup (n \sim S Z) \n\} 

inconsistent
continued

\[
\{ (VC \_ \_ ) \_ \_ \n\rightarrow n \sim S m \}
\]

\[
(VC \ x \ xs) \ (VC \ y \ ys)
\]

\[
\{ (VC \_ \_ ) \ (VC \_ \_ ) \n\rightarrow (n\sim S m) \cup (n\sim S o) \}
\]

\[
\{ (VC \_ \_ ) \n\rightarrow n\sim S m \}
\]

\[
\{ (VC \_ \_ ) VN \n\rightarrow (n \sim S m) \cup (n \sim Z) \}
\]

inconsistent
continued

\{ (VC _ _) _ \uparrow \ n \sim S \ m \}\n
\{ (VC x xs) (VC y ys) \n
\{ (VC _ _) (VC _ _) \uparrow \ (n\sim S \ m)\cup(n\sim S \ o) \n\{ (VC _ _) \ \bot \uparrow \ n\sim S \ m \}
continued

\{ (VC _ _) _ \uparrow n \sim S \ m \}

\{ (VC _ _) (VC _ _) \uparrow (n \sim S \ m) \cup (n \sim S \ o) \}

\{ (VC _ _) \downarrow \uparrow n \sim S \ m \}

\emptyset

exhaustive!
suppose one more catch-all clause
suppose one more catch-all clause
suppose one more catch-all clause
Guards
Guards

\[
\begin{align*}
\text{abs } x & \quad | \quad x < 0 & = & \ldots \\
& \quad | \quad x \geq 0 & = & \ldots 
\end{align*}
\]
Guard Core Syntax

**Pattern**

\[ p ::= x \]  
\[ | \quad k \vec{p} \]  
\[ | \quad p <-- e \]

variable names  
pattern guards

\[ \text{abs } x \ (\text{True }<- x < 0) = \ldots \]
\[ \text{abs } x \ (\text{True }<- x >= 0) = \ldots \]
More Desugaring

literal patterns

\[
f :: \text{Num} \ a \Rightarrow a \rightarrow [b] \rightarrow \ldots \\
f \ 0 \ \text{[]} = \ \ldots
\]
More Desugaring

literal patterns

\[ f :: \text{Num} \ a \Rightarrow a \rightarrow \ [b] \rightarrow \ldots \]
\[ f \ 0 \ [] = \ldots \]
\[ f \ x \ (\text{True} \leftarrow x == \text{fromInteger} \ 0) \ [] = \ldots \]
More Desugaring

literal patterns

\[ f :: \text{Num} \ a \to \ a \to \ [b] \to \ldots \]
\[ f \ 0 \ [ ] = \ldots \]
\[ f \ x \ (\text{True} \leftarrow x = \text{fromInteger} \ 0) \ [ ] = \ldots \]

view patterns

\[ g \ (\text{toRad} \to (\text{theta}, r)) = \ldots \]
More Desugaring

**literal patterns**

\[
\begin{align*}
f & : \text{Num a ==> a --> [b] --> ...} \\
f & 0 \ [\ ] = \ ... \\
f & x \ (\text{True <- x == fromInteger 0}) \ [\ ] = \ ...
\end{align*}
\]

**view patterns**

\[
\begin{align*}
g & \ (\text{toRad --> (theta,r)}) = \ ...
\end{align*}
\]

\[
\begin{align*}
g & \ x \ ((\text{theta},r) \ <- \ \text{toRad} \ x) = \ ...
\end{align*}
\]
Guarded Value Abstraction

Value Abstraction

\( v ::= \Gamma \vdash \bar{u} \triangleright \Delta \)

\( u ::= x \mid K \bar{u} \)

\( \Gamma ::= \epsilon \mid \Gamma, a \mid \Gamma, x : \tau \)

\( \Delta ::= \epsilon \mid \Delta \cup \Delta \mid \tau \sim \tau \)

| \quad x \approx e |
| \quad x \approx \bot |
x \ (\text{True} \leftarrow x<0)
\[ \{ \text{__}\text{__} \triangleright \varepsilon \} \]

x \ (\text{True} \leftarrow x<0)
unconstrained

\{ \_ \_ \_ \triangleright \varepsilon \}\n
x (True <- x<0)
\[\{__ __ \triangleright \varepsilon\}\]

\[x \ (\text{True} \leftarrow x < 0) \]

\[\{x \triangleright y \approx (x < 0) \bigcup y \approx \text{True}\}\]

unconstrained
\[\{x \Rightarrow y \approx (x < 0) \cup y \approx \text{True}\}\]

\[\{x \Rightarrow y \approx (x < 0) \cup y \approx \bot\}\]
unconstrained

\{\_ \_ \_ \_ \_ \varepsilon\}

\begin{align*}
\text{x (True }&\text{ <- x<0) } \\
\{&\text{x }\uparrow \text{ y≈(x<0) } \\
&\text{U y≈True}\} \\
\{&\text{x }\uparrow \text{ y≈(x<0) } \\
&\text{U y≈⊥}\} \\
\{&\text{x }\uparrow \text{ y≈(x<0) U y≈False}\}
\end{align*}
continued

\[ x \ (True \leftarrow x \geq 0) \]
continued

\{\_ \_ \uparrow x \uparrow y \approx (x<0) \cup y \approx \text{False}\}

x \ (\text{True } \leftarrow \ x \geq 0)
continued

\{ _ _ \triangleright x \triangleright y \approx (x < 0) \cup y \approx False\}

\{ x \triangleright y \approx (x < 0) \cup y \approx False \cup z \approx (x \geq 0) \cup z \approx True \}

x (True \leftarrow x \geq 0)
continued

\{ \_ \_ \triangleleft x \triangleleft y \approx (x<0) \cup y \approx \text{False} \}

\{ x \triangleleft y \approx (x<0) \\
U y \approx \text{False} \\
U z \approx (x\geq0) \\
U z \approx \bot \}

x \ (\text{True} \leftarrow x\geq0)
continued

\{ _ _ \triangleright x \triangleright y\approx(x<0) \cup y\approx\text{False}\}

\{x \triangleright y\approx(x<0) \\
\cup y\approx\text{False} \\
\cup z\approx(x\geq0) \\
\cup z\approx\text{True}\}

x (\text{True} \leftarrow x\geq0)
continued

\[ \{ x \triangleright y \approx (x < 0) \cup y \approx \text{False} \} \]

\[ \{ x \triangleright y \approx (x < 0) \cup y \approx \text{False} \cup z \approx (x \geq 0) \cup z \approx \text{True} \} \]

\[ x \ (\text{True } \leftarrow x \geq 0) \]

\[ \emptyset \]
continued

\{\_\_ \triangleright x \triangleright y \equiv (x<0) \cup y \equiv \text{False}\}

\{x \triangleright y \equiv (x<0) \cup y \equiv \text{False} \cup z \equiv (x\geq 0) \cup z \equiv \text{True}\}

x \ (\text{True} \leftarrow x \geq 0)

\emptyset

\emptyset
Evaluation
Prototype Implementation

- GHC branch
- 504 LoC (vs. old 588 LoC)
- Type oracle: GHC type checker
- Term oracle: for now only \((y \approx \text{False} \cup y \approx \text{False})\)
### Hackage Packages

<table>
<thead>
<tr>
<th>Hackage Packages</th>
<th>LoC</th>
<th>GHC-1 M</th>
<th>GHC-1 R</th>
<th>GHC-2 M</th>
<th>GHC-2 R</th>
<th>New M</th>
<th>New R</th>
</tr>
</thead>
<tbody>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,008</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>hoopl</td>
<td>2,147</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>json-sop</td>
<td>393</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>lens-sop</td>
<td>280</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>pretty-sop</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Additional tests

<table>
<thead>
<tr>
<th>Additional tests</th>
<th>LoC</th>
<th>GHC-1 M</th>
<th>GHC-1 R</th>
<th>GHC-2 M</th>
<th>GHC-2 R</th>
<th>New M</th>
<th>New R</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hackage Packages</td>
<td>GHC-1 LoC</td>
<td>GHC-1 M</td>
<td>GHC-1 R</td>
<td>GHC-2 M</td>
<td>GHC-2 R</td>
<td>New M</td>
<td>New R</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>accelerate</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>ad</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,008</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>hoopl</td>
<td>2,147</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>json-sop</td>
<td>393</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>lens-sop</td>
<td>280</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>pretty-sop</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>Additional tests</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Hackage Packages</td>
<td>LoC</td>
<td>M</td>
<td>R</td>
<td>LoC</td>
<td>M</td>
<td>R</td>
<td>LoC</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>-----</td>
<td>----</td>
<td>----</td>
<td>-----</td>
</tr>
<tr>
<td>accelerate</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>ad</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,008</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>hoopl</td>
<td>2,147</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>json-sop</td>
<td>393</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>lens-sop</td>
<td>280</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>pretty-sop</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional tests</th>
<th>LoC</th>
<th>M</th>
<th>R</th>
<th>LoC</th>
<th>M</th>
<th>R</th>
<th>LoC</th>
<th>M</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hackage Packages</td>
<td>GHC-1</td>
<td>GHC-2</td>
<td>New</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>accelerate</td>
<td>116</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ad</td>
<td>106</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,008</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hoopl</td>
<td>2,147</td>
<td>33</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>json-sop</td>
<td>393</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lens-sop</td>
<td>280</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pretty-sop</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional tests</th>
<th>LoC</th>
<th>M</th>
<th>R</th>
<th>M</th>
<th>R</th>
<th>M</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hackage Packages</td>
<td>LoC</td>
<td>M</td>
<td>R</td>
<td>LoC</td>
<td>M</td>
<td>R</td>
<td>LoC</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------</td>
<td>----</td>
<td>----</td>
<td>------</td>
<td>----</td>
<td>----</td>
<td>------</td>
</tr>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>generics-sop</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hoopl</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>json-sop</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lens-sop</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pretty-sop</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional tests</th>
<th>R</th>
<th>M</th>
<th>R</th>
<th>M</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
## Evaluation

We compared three checkers. The baseline is, of course, vanilla GHC. The authors found the GHC-2 checker to be superior to the baseline by a significant margin. Of interest is the results of how effective the warning reports for GADTs, so we ran GHC two ways: both with and without ad-hoc hack of GHC-2 was quite successful at eliminating unnecessary clauses. We observed for satisfiability, inference of type equality constraints can be flushed out by moving the equality witness to the front of the constructor.

### Table 1

<table>
<thead>
<tr>
<th>Hackage Packages</th>
<th>LoC</th>
<th>GHC-1 M</th>
<th>GHC-1 R</th>
<th>GHC-2 M</th>
<th>GHC-2 R</th>
<th>New M</th>
<th>New R</th>
</tr>
</thead>
<tbody>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,903</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>hoopl</td>
<td>273</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>json-sop</td>
<td>393</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>lens-sop</td>
<td>92</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>pretty-sop</td>
<td>77</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hackage Packages</td>
<td>LoC</td>
<td>GHC-1 M</td>
<td>GHC-1 R</td>
<td>GHC-2 M</td>
<td>GHC-2 R</td>
<td>New M</td>
<td>New R</td>
</tr>
<tr>
<td>----------------------</td>
<td>------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,293</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>hoopl</td>
<td>1,377</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>json-sop</td>
<td>1,217</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>lens-sop</td>
<td>1,353</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>pretty-sop</td>
<td>1,277</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional tests</th>
<th>LoC</th>
<th>GHC-1 M</th>
<th>GHC-1 R</th>
<th>GHC-2 M</th>
<th>GHC-2 R</th>
<th>New M</th>
<th>New R</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Clauses erroneously reported: Missing
We compared three checkers. The baseline is, of course, vanilla GHC-1: it is conservative, and reports too many missing clauses; whereas GHC-2 is conservative, and reports too few redundant clauses; so a higher degree of pessimism is needed to be checked, we can instead pass the state to the solver.

In the process of checking given constraints for satisfiability, the solver believes the constraints are satisfiable, it returns their normal form. When later the conjunction of these constraints by substituting for equality of the constructor.

We have found three cases are not covered by the approach was compared with our new algorithm for Section 1.

We have paraphrased one such example in terms of the following type of Section 2: GADTs, laziness, and guards. However in our evaluation, we cannot possibly have an equality witness for equality of.

This example uses the ad-hoc hack of GHC-2 was quite successful at eliminating unnecessary.

Additional tests

<table>
<thead>
<tr>
<th>Hackage Packages</th>
<th>LoC</th>
<th>GHC-1</th>
<th>GHC-2</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,993</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hoopl</td>
<td>477</td>
<td>33</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>json-sop</td>
<td>79</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lens-sop</td>
<td>86</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pretty-sop</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional tests</th>
<th>LoC</th>
<th>GHC-1</th>
<th>GHC-2</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

We have also been able to greatly save on solving time. Moreover, by finding inconsistencies for GADT-using programs. This has resulted in identifying 9 close nine GHC tickets related to GADT pattern matching (#3927, #256, #903, #27, #111, #2753, #1993, #477, #79, #86, #67).

<table>
<thead>
<tr>
<th>Hackage Packages</th>
<th>LoC</th>
<th>GHC-1</th>
<th>GHC-2</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,993</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hoopl</td>
<td>477</td>
<td>33</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>json-sop</td>
<td>79</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lens-sop</td>
<td>86</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pretty-sop</td>
<td>67</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional tests</th>
<th>LoC</th>
<th>GHC-1</th>
<th>GHC-2</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hackage Packages</td>
<td>LoC</td>
<td>GHC-1 M</td>
<td>GHC-1 R</td>
<td>GHC-2 M</td>
</tr>
<tr>
<td>----------------------</td>
<td>------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,933</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hoopl</td>
<td>77</td>
<td>33</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>json-sop</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lens-sop</td>
<td>74</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pretty-sop</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Additional tests</strong></td>
<td></td>
<td><strong>M</strong></td>
<td><strong>R</strong></td>
<td><strong>M</strong></td>
</tr>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Evaluation

Because the algorithm refines step by step one initial value to a solution, we can instead pass the state of the solver. Because the solver can be checked, we can instead pass the state of the oracle.

The number of missing clauses (\(M\)).

<table>
<thead>
<tr>
<th>Hackage Packages</th>
<th>LoC</th>
<th>M</th>
<th>R</th>
<th>GHC-1</th>
<th>GHC-2</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,228</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hooipl</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>json-sop</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lens-sop</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pretty-sop</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional tests</th>
<th>LoC</th>
<th>M</th>
<th>R</th>
<th>GHC-1</th>
<th>GHC-2</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **clauses reported**: Redundant
### Results

<table>
<thead>
<tr>
<th>Hackage Packages</th>
<th>LoC</th>
<th>M</th>
<th>R</th>
<th>M</th>
<th>R</th>
<th>M</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,993</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>hoopl</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>json-sop</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>lens-sop</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>pretty-sop</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional tests</th>
<th>LoC</th>
<th>M</th>
<th>R</th>
<th>M</th>
<th>R</th>
<th>M</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Hackage Packages</td>
<td>LoC</td>
<td>M</td>
<td>R</td>
<td>M</td>
<td>R</td>
<td>M</td>
<td>R</td>
</tr>
<tr>
<td>------------------------</td>
<td>------</td>
<td>----</td>
<td>---</td>
<td>----</td>
<td>---</td>
<td>----</td>
<td>---</td>
</tr>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,928</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>hoopl</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>json-sop</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>lens-sop</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>pretty-sop</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional tests</th>
<th>LoC</th>
<th>M</th>
<th>R</th>
<th>M</th>
<th>R</th>
<th>M</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The table shows the number of redundant clauses reported by GHC versions 1 and 2, as well as a new approach. GHC-1 is shown to report too many redundant clauses, while GHC-2 and the new approach report an appropriate number of redundant clauses.
<table>
<thead>
<tr>
<th>Hackage Packages</th>
<th>LoC</th>
<th>M</th>
<th>GHC-1</th>
<th>GHC-2</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>accelerate</td>
<td>11,393</td>
<td>11</td>
<td>0</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>ad</td>
<td>1,903</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>boolsimplifier</td>
<td>256</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d-bus</td>
<td>2,753</td>
<td>45</td>
<td>0</td>
<td>42</td>
<td>16</td>
</tr>
<tr>
<td>generics-sop</td>
<td>1,992</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>hoopl</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>json-sop</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>lens-sop</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>pretty-sop</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>Additional tests</strong></td>
<td>LoC</td>
<td>M</td>
<td>GHC-1</td>
<td>GHC-2</td>
<td>New</td>
</tr>
<tr>
<td>lists</td>
<td>66</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>heterogeneous lists</td>
<td>38</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>heaps</td>
<td>540</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Space Explosion?

<table>
<thead>
<tr>
<th>Maximum size of C/U</th>
<th>Pattern Matches</th>
<th>(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>8702</td>
<td>97.90</td>
</tr>
<tr>
<td>10 – 99</td>
<td>181</td>
<td>2.04</td>
</tr>
<tr>
<td>100 – 2813</td>
<td>5</td>
<td>0.06</td>
</tr>
</tbody>
</table>
## Space Explosion?

<table>
<thead>
<tr>
<th>Maximum size of C/U</th>
<th>Pattern Matches</th>
<th>(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 9</td>
<td>8702</td>
<td>97.90</td>
</tr>
<tr>
<td>10 - 99</td>
<td>181</td>
<td>2.04</td>
</tr>
<tr>
<td>100 - 2813</td>
<td>5</td>
<td>0.06</td>
</tr>
</tbody>
</table>

maximum size of Covered/Unconvered sets per match
Space Explosion?

<table>
<thead>
<tr>
<th>Maximum size of C/U</th>
<th>Pattern Matches</th>
<th>(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>8702</td>
<td>97.90</td>
</tr>
<tr>
<td>10 – 99</td>
<td>181</td>
<td>2.04</td>
</tr>
<tr>
<td>100 – 2813</td>
<td>5</td>
<td>0.06</td>
</tr>
</tbody>
</table>
Space Explosion?

<table>
<thead>
<tr>
<th>Maximum size of C/U</th>
<th>Pattern Matches</th>
<th>(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>8702</td>
<td>97.90</td>
</tr>
<tr>
<td>10 – 99</td>
<td>181</td>
<td>2.04</td>
</tr>
<tr>
<td>100 – 2813</td>
<td>5</td>
<td>0.06</td>
</tr>
</tbody>
</table>
Space Explosion?

<table>
<thead>
<tr>
<th>Maximum size of C/U</th>
<th>Pattern Matches</th>
<th>(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>8702</td>
<td>97.90</td>
</tr>
<tr>
<td>10 – 99</td>
<td>181</td>
<td>2.04</td>
</tr>
<tr>
<td>100 – 2813</td>
<td>5</td>
<td>0.06</td>
</tr>
</tbody>
</table>
Space Explosion?

<table>
<thead>
<tr>
<th>Maximum size of C/U</th>
<th>Pattern Matches</th>
<th>(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>8702</td>
<td>97.90</td>
</tr>
<tr>
<td>10 – 99</td>
<td>181</td>
<td>2.04</td>
</tr>
<tr>
<td>100 – 2813</td>
<td>5</td>
<td>0.06</td>
</tr>
</tbody>
</table>

all in package ad

data T = A | B | C
f A A = True
f B B = True
f C C = True
Space Explosion?

<table>
<thead>
<tr>
<th>Maximum size of C/U</th>
<th>Pattern Matches</th>
<th>(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9</td>
<td>8702</td>
<td>97.90</td>
</tr>
<tr>
<td>10 – 99</td>
<td>181</td>
<td>2.04</td>
</tr>
<tr>
<td>100 – 2813</td>
<td>5</td>
<td>0.06</td>
</tr>
</tbody>
</table>

54 x 54 constructors

all in package ad

data T = A | B | C
f A A = True
f B B = True
f C C = True
A uniform approach to deal with:

- GADTs
- Laziness
- Guards and extensions
  pattern guards, view patterns, literal patterns, …
More in the Paper

- Full Algorithm
- Optimisation Opportunities
- Meta-Theory
- Related Work
The End