WIP - Caching constraints during type checking

Bohdan Liesnikov and Jesper Cockx

27th November 2024

Delft University of Technology

Things you already know

Instance search

```
_==_ : {A : Set} {{eqA : Eq A}} → A → A → Bool

elem : {A : Set} {{eqA : Eq A}} → A → List A → Bool
elem x (y ∷ xs) = x == y || elem x xs
elem x []       = false
```

Agda’s instance search

- was backtracking by default, not anymore
- reimplemented and sped up

Backtracking

- Causes exponential blow-up [Isa18].
- Disabled by Jesper[Coc18] in 2018.

New implementation

Amy's implementation [Lia24a; Lia24b]

- Based on discrimination trees
- Finds one instance at a time

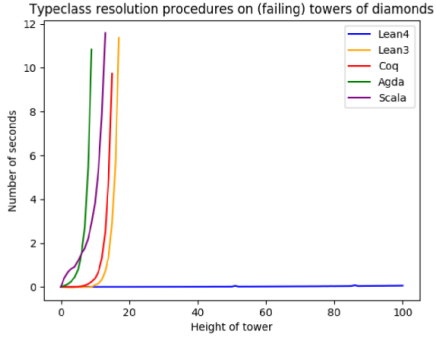
New implementation

Profiling point	2.6.4.3	2.6.5
Total	703,384ms	693,122ms
Typing	48,922ms	50,900ms
Typing.InstanceSearch	918ms	852ms
Typing.InstanceSearch.FilterCandidates	64,238ms	34,063ms
Typing.InstanceSearch.InitialCandidates	1,954ms	7,546ms
Typing.OccursCheck	44,405ms	30,683ms

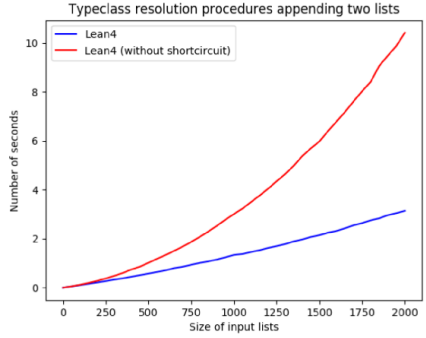
Lean's implementation

- "tabled resolution" [SUdM20]
- eliminates exponential blowup
- keeps backtracking

Tabled typeclass resolution



(a)



(b)

Goal

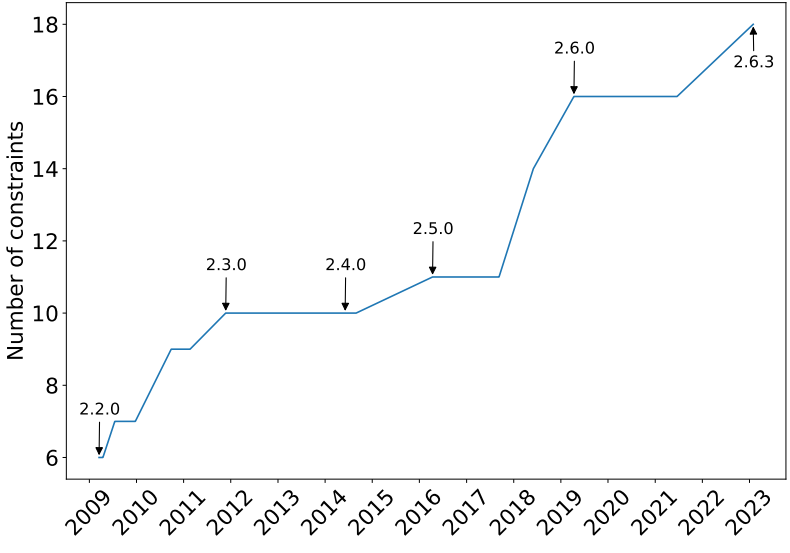
Get “caching” for instance search problems

Generalise!

Goal

Re-use the solutions to constraints that appear more than once

Agda's constraints



One more added since then

Agda's constraints

```
data Constraint
  = ValueCmp Comparison CompareAs Term Term
  | ValueCmpOnFace Comparison Term Type Term Term
  | ElimCmp [Polarity] [IsForced] Type Term [Elim] [Elim]
  | SortCmp Comparison Sort Sort
  | LevelCmp Comparison Level Level
  | HasBiggerSort Sort
  | HasPTSRule (Dom Type) (Abs Sort)
  | CheckDataSort QName Sort
  | CheckMetaInst MetaId
  | CheckType Type
  | UnBlock MetaId
  | IsEmpty Range Type
  | CheckSizeLtSat Term
  | FindInstance MetaId (Maybe [Candidate])
  | ResolveInstanceHead QName
  | CheckFunDef A.DefInfo QName [A.Clause] TCErr
  | UnquoteTactic Term Term Type
  | CheckLockedVars Term Type (Arg Term) Type
  | UsableAtModality WhyCheckModality (Maybe Sort) Modality Term
```

Disqualifying constraints

```
data Constraint
  = CheckDataSort QName Sort
  | CheckMetaInst MetaId
  | CheckType Type
  | UnBlock MetaId
  | CheckSizeLtSat Term
  | CheckFunDef A.DefInfo QName [A.Clause] TCErr
  | UnquoteTactic Term Term Type
  | CheckLockedVars Term Type (Arg Term) Type
```

Qualifying constraints

```
data Constraint
  = ValueCmp Comparison CompareAs Term Term
  | ValueCmpOnFace Comparison Term Type Term Term
  | ElimCmp [Polarity] [IsForced] Type Term [Elim] [Elim]
  | SortCmp Comparison Sort Sort
  | LevelCmp Comparison Level Level
  | HasPTSRule (Dom Type) (Abs Sort)
  | IsEmpty Range Type
  | FindInstance MetaId (Maybe [Candidate])
  | ResolveInstanceHead QName
  | UsableAtModality WhyCheckModality (Maybe Sort) Modality Term
```

Qualifying constraints

```
data Constraint
  = ValueCmp
  | ValueCmpOnFace
  | ElimCmp
  | SortCmp
  | LevelCmp
  | HasPTSRule
  | IsEmpty
  | FindInstance
  | ResolveInstanceHead
  | UsableAtModality
```

Complications

What does it mean to be the same?

Numbers

Approximations

```
+ substitute all vars for zeros, throw away the context

~ substitute all vars for zeros, keep the context
* constraints that can be cached
~ same terms, throw away the context

- same terms, keep the context
```

Example

agda/s/s/Data/Bool/**Properties.agda**

```
1  ------------------------------------------------------------------------
2  -- The Agda standard library
3  --
4  -- A bunch of properties
5  ------------------------------------------------------------------------
6
7  {-# OPTIONS --cubical-compatible --safe #-}
8
9  module Data.Bool.Properties where
10
```

FIGURE 1: agda.github.io/agda-stdlib/v2.0/Data.Bool.Properties

Approximation from above

Constraint type	Count	Constraint
ElimCmp	344	[] ~~ [] : Set
ValueCmp	92	Agda.Builtin.Bool.Bool -> Agda.Builtin.Bool.Bool -> Agc
ElimCmp	33	[] ~~ [] : Set lzero
ElimCmp	31	[Data.Empty.Empty] ~~ [Data.Empty.Empty] : (A : Set l
ElimCmp	31	[{lzero}, Data.Empty.Empty] ~~ [{lzero}, Data.Empty.Em
ElimCmp	25	[] ~~ [] : Agda.Builtin.Bool.Bool
ElimCmp	25	[] ~~ [] : Agda.Builtin.Bool.Bool
ElimCmp	24	[] ~~ [] : Agda.Builtin.Bool.Bool -> Agda.Builtin.Bool.Bo
ElimCmp	18	[] ~~ [] : Set lzero
ElimCmp	14	[] ~~ [] : Agda.Builtin.Bool.Bool -> Agda.Builtin.Bool.Bo
ElimCmp	13	[] ~~ [] : Agda.Builtin.Bool.Bool
ValueCmp	11	Agda.Builtin.Unit.⊤ = Data.Bool.Base.T Agda.Builtin.B
ValueCmp	11	Data.Irrelevant.Irrelevant {lzero} Data.Empty.Empty = D
ValueCmp	11	Agda.Builtin.Bool.Bool -> Agda.Builtin.Bool.Bool =< Alg
HasPTSRule	9	Has PTS rule: ({Agda.Builtin.Bool.Bool} , Set)
ElimCmp	8	[] ~~ [] : {a : Agda.Primitive.Level} -> {A : Set @0} -> @
HasPTSRule	7	Has PTS rule: ({Agda.Builtin.Bool.Bool} , Set)
ElimCmp	6	[] ~~ [] : Agda.Builtin.Bool.Bool
ElimCmp	6	[@0] ~~ [@0] : Agda.Builtin.Bool.Bool -> Agda.Builtin.B
HasPTSRule	6	Has PTS rule: ((Agda.Builtin.Bool.Bool) , Set)

Figure 2: Zero all vars and drop context

Approximation from below

Constraint type	Count	Constraint	Context
ElimCmp	227	[] ~~ [] : Set	[]
ValueCmp	92	Agda.Builtin.Bool.Bool -> Agda.Builtin.Bool.Bool -> Agda.Bui	[]
ElimCmp	17	[] ~~ [] : Agda.Builtin.Bool.Bool	[]
ElimCmp	15	[] ~~ [] : Agda.Builtin.Bool.Bool	[]
ValueCmp	11	Agda.Builtin.Bool.Bool -> Agda.Builtin.Bool.Bool =< Algebra.	[]
ElimCmp	8	[] ~~ [] : Agda.Builtin.Bool.Bool -> Agda.Builtin.Bool.Bool -> S	[]
HasPTSRule	6	Has PTS rule: ({Agda.Builtin.Bool.Bool} , funSort Set Set)	[]
SortCmp	5	Set =< Set	[]
ElimCmp	4	[] ~~ [] : (dummyType: __DUMMY_TYPE__, called at src/full	[]
ElimCmp	4	[] ~~ [] : (dummyType: __DUMMY_TYPE__, called at src/full	[]
HasPTSRule	4	Has PTS rule: ((Agda.Builtin.Bool.Bool) , funSort Set Set)	[]
ValueCmp	3	(x : Agda.Builtin.Bool.Bool) -> (y : Agda.Builtin.Bool.Bool) ->	[]
ValueCmp	3	(x : Agda.Builtin.Bool.Bool) -> (y : Agda.Builtin.Bool.Bool) ->	[]
ElimCmp	3	[] ~~ [] : Agda.Builtin.Bool.Bool -> Agda.Builtin.Bool.Bool -> S	[]
ElimCmp	3	[] ~~ [] : Set lzero	[]
ValueCmp	2	Data.Bool.Base._∧_ Agda.Builtin.Bool.Bool.false Agda.Built	[]
ValueCmp	2	Data.Bool.Base._∧_ Agda.Builtin.Bool.Bool.true Agda.Builti	[]
ValueCmp	2	Data.Bool.Base._∨_ Agda.Builtin.Bool.Bool.false Agda.Built	[]
ValueCmp	2	Data.Bool.Base._∨_ Agda.Builtin.Bool.Bool.true Agda.Builti	[]
ValueCmp	2	Data.Bool.Base._xor_ Agda.Builtin.Bool.Bool.false Agda.Bui	[]
ValueCmp	2	Data.Irrelevant.Irrelevant {lzero} Data.Empty.Empty = Data.E	[]
ValueCmp	2	Agda.Builtin.Equality._≡_ {lzero} {Agda.Builtin.Bool.Bool} (Da	[]
ValueCmp	2	Agda.Builtin.Sigma.Σ {lub lzero lzero} {lub lzero lzero} (Algeb	[]
ValueCmp	2	Agda.Builtin.Sigma.Σ {lub lzero lzero} {lub lzero lzero} (Algeb	[]

FIGURE 3: Keep variables and context

Example 2

agda/std-lib/src/Data/Integer/**Properties.agda**

```
1  ------------------------------------------------------------------------
2  -- The Agda standard library
3  --
4  -- Some properties about integers
5  ------------------------------------------------------------------------
6
7  {-# OPTIONS --cubical-compatible --safe #-}
8
9  module Data.Integer.Properties where
10
11 open import Algebra.Bundles
12 import Algebra.Morphism as Morphism
13 open import Algebra.Construct.NaturalChoice.Base
14 import Algebra.Construct.NaturalChoice.MinMaxOp as MinMaxOp
15 import Algebra.Lattice.Construct.NaturalChoice.MinMaxOp as LatticeMinMaxOp
```

Figure 4: agda.github.io/agda-stdlib/v2.0/Data.Integer.Properties

Approximation from above

Constraint type	Count	Constraint
SortCmp	1499	Set =< Set
ElimCmp	782	[] ~~ [] : Set
ElimCmp	684	[] ~~ [] : Set
HasPTSRule	656	Has PTS rule: ((Agda.Builtin.Nat.Nat) , Set)
ValueCmp	544	Set =< Set
HasPTSRule	366	Has PTS rule: ((Agda.Builtin.Int.Int), Set)
ElimCmp	268	[] ~~ [] : {a : Agda.Primitive.Level} -> {A : Set @0}
ElimCmp	206	[] ~~ [] : Agda.Builtin.Int.Int -> Agda.Builtin.Int.Int -
LevelCmp	197	lzero =< lzero
HasPTSRule	163	Has PTS rule: ((Data.Nat.Base._≤_ @0 @0), Set)
HasPTSRule	161	Has PTS rule: ((Agda.Builtin.Int.Int) , Set)
HasPTSRule	145	Has PTS rule: ((Agda.Builtin.Nat.Nat) , Set)
ElimCmp	132	[] ~~ [] : Agda.Builtin.Int.Int -> Agda.Builtin.Int.Int -
HasPTSRule	119	Has PTS rule: ((Data.Nat.Base._≤_ @0 @0 -> Da
HasPTSRule	115	Has PTS rule: ((Agda.Builtin.Int.Int) , Set)
HasPTSRule	114	Has PTS rule: ((Data.Nat.Base._≤_ @0 @0) , Set)
HasPTSRule	108	Has PTS rule: ((Agda.Builtin.Equality._≡_ {lzero} {
HasPTSRule	98	Has PTS rule: ((Agda.Builtin.Nat.Nat) , Set)
HasPTSRule	96	Has PTS rule: ({Agda.Builtin.Int.Int} , Set)
HasPTSRule	91	Has PTS rule: ({Agda.Builtin.Int.Int} , Set)
HasPTSRule	83	Has PTS rule: ((Data.Integer.Base._<_ @0 @0), S
HasPTSRule	75	Has PTS rule: ((Data.Integer.Base._≤_ @0 @0), S
ElimCmp	66	[] ~~ [] : Agda.Builtin.Nat.Nat -> Agda.Builtin.Nat.N
ValueCmp	64	Set Level.0ℓ =< Set
SortCmp	64	Set Level.0ℓ =< Set

FIGURE 5: Zero all vars and drop context

Next steps

- caching for conservative equality
- hashing modulo alpha-equivalence [BOG24]

Problems

- Localise Blaauwbroek's algorithm
- What is a suitable granularity for caching?
- What are the conditions for solution re-use?

[BOG24] Lasse Blaauwbroek, Miroslav Olšák, and Herman Geuvers. **“Hashing Modulo Context-Sensitive ⊠-Equivalence”.** In: *Proceedings of the ACM on Programming Languages* 8 (PLDI June 20, 2024), 229:2027–229:2050. DOI: 10.1145/3656459. URL: https://dl.acm.org/doi/10.1145/3656459 (visited on 06/21/2024).

[Coc18] Jesper Cockx. ***Allow Unconstrained Instances & Disallow Overlapping Instances by Jespercockx · Pull Request #3419 · Agda/Agda*.** GitHub. Nov. 29, 2018. URL: https://github.com/agda/agda/pull/3419 (visited on 11/26/2024).

[Isa18] Isaac Elliott. ***Quadratic (Failing) Instance Search · Issue #2993 · Agda/Agda*.** GitHub. Mar. 8, 2018. URL: https://github.com/agda/agda/issues/2993 (visited on 11/26/2024).

[Lia24a] Amélia Liao. ***Discrimination Trees for Instance Search by Plt-Amy · Pull Request #7109 · Agda/Agda*.** GitHub. Feb. 12, 2024. URL: https://github.com/agda/agda/pull/7109 (visited on 11/26/2024).

[Lia24b] Amélia Liao. ***Efficient Instance Resolution for Agda*.** Amélia's blog. Mar. 8, 2024. URL: https://amelia.how/posts/efficient-instance-resolution-for-agda.html (visited on 11/26/2024).

[SUdM20] Daniel Selsam, Sebastian Ullrich, and Leonardo de Moura. **"Tabled Typeclass Resolution".** Jan. 21, 2020. DOI: 10.48550/arXiv.2001.04301. arXiv: 2001.04301 [cs]. URL: http://arxiv.org/abs/2001.04301 (visited on 06/07/2022).