Solving Clafer Models with Choco

Jia (Jimmy) Liang

University of Waterloo

December 17, 2012

Clafer with Alloy

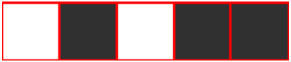
- Translation from Clafer to Alloy is straight forward
- Semantically similar: sets, joins, constraints
- Clafer corresponds to sig with additional constraints to enforce structure
- Integers are handled by bit blasting

Choco

- Constraint programming library written in Java
- 4 types of variables
  - Integer variable - Java integers
  - Set variable - Sets over integers
  - Real variable - Java double
  - ~~Task variable~~
- AC3, AC3rm, AC2001, AC2008, etc.
- Single objective
- Custom constraints

Notation

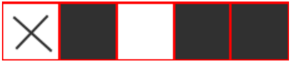
Set



$$set \subseteq \{0,1,2,3,4\}$$

$$set = \{1,3,4\}$$

$$|set| = \text{number of filled in slots} = 3$$



$$set \subseteq \{1,2,3,4\}$$

$$set = \{1,3,4\}$$

Notation

Integer

23	43	22	0	1

$$
\begin{aligned}
intarray &\in \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \\
intarray &= [23, 43, 22, 0, 1] \\
\mathbb{Z}_n &= \{0, 1, ..., n\} \\
\mathbb{Z}_{m,n} &= \{m, m+1, ..., n\}
\end{aligned}
$$

Simple model

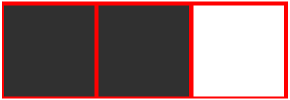
Clafer 1: Cats

```
Cat 2
    Eye 2
```

$$scope(Cat) = 3, scope(Eye) = 6$$

Parent

$$Cat \subseteq \{0, 1, 2\}$$



Clafer 2: Cats

```
Cat0
Cat1
```

Parent-child

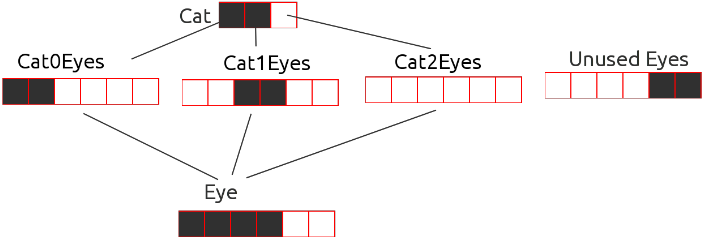
$$Cat \subseteq \{0, 1, 2\}$$

$$Cat_0Eyes, Cat_1Eyes, Cat_2Eyes, UnusedEyes \subseteq \{0, 1, 2, 3, 4, 5\}$$

$$Eye = \subseteq \{0, 1, 2, 3, 4, 5\}$$

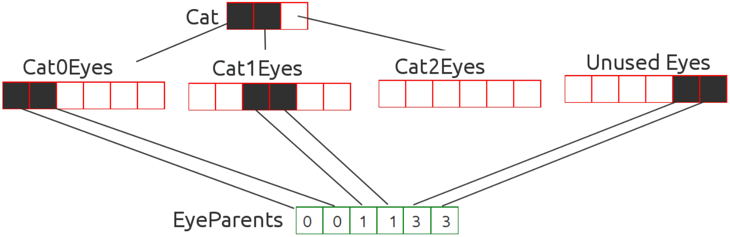
$$\bigwedge_{i=0}^{2} (if \;\; i \in Cat \;\; then \; |Cat_iEyes| = 2 \;\; else \;\; Cat_iEyes = \emptyset)$$

$$Eye = Cat_0Eyes \cup Cat_1Eyes \cup Cat_2Eyes$$



Parent pointers

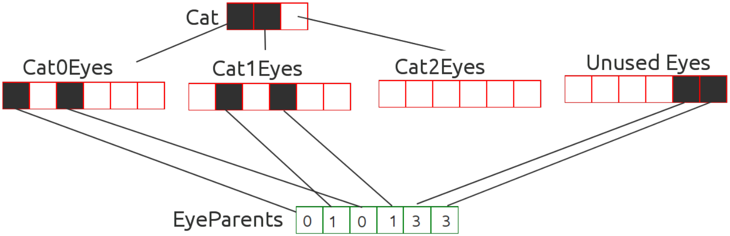
$$EyeParents \in \mathbb{Z}_3 \times \mathbb{Z}_3 \times \mathbb{Z}_3 \times \mathbb{Z}_3 \times \mathbb{Z}_3 \times \mathbb{Z}_3$$



From official documentation: **inverseSet**$(\langle x_1, .., x_n\rangle, \langle y_1, .., y_m\rangle)$ states that x_i has value j if and only if y_j contains value i:

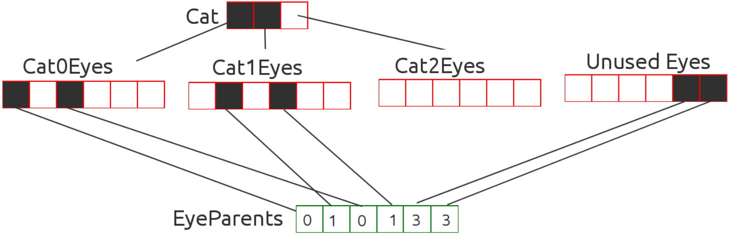
$$x_i = j \iff i \in y_j, \forall i = 0...n-1, j = 0...m-1$$

Isomorphism



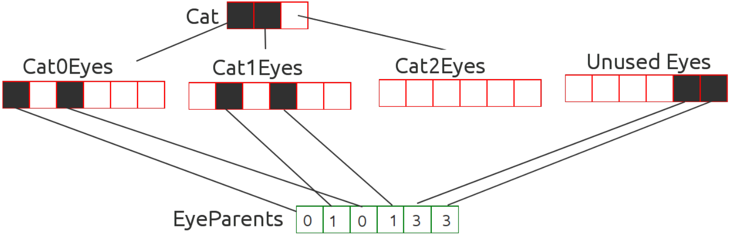
- 270 solutions, all isomorphic

Isomorphism



- 270 solutions, all isomorphic
- Force EyeParents to be sorted, 1 solution

Isomorphism

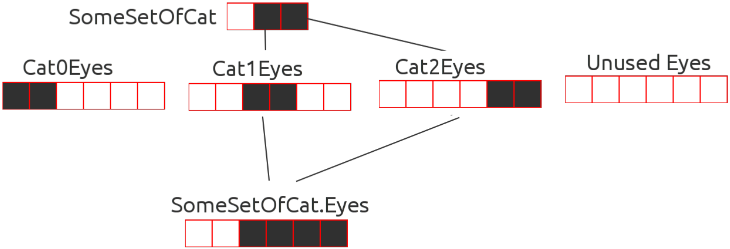


- 270 solutions, all isomorphic
- Force EyeParents to be sorted, 1 solution
- Shrink the bounds of $Cat_i Eyes$

$$
\begin{aligned}
Cat_0 Eyes &\subseteq \{0, 1\} \\
Cat_1 Eyes &\subseteq \{2, 3\} \\
Cat_2 Eyes &\subseteq \{4, 5\}
\end{aligned}
$$

Joining with children

How to implement the expression *SomeSetOfCat*.*Eye*?
Suppose *SomeSetOfCat* is a subexpression that evaluates to $\{Cat_1, Cat_2\}$.



Joining with inherited children

Clafer 3: Animals

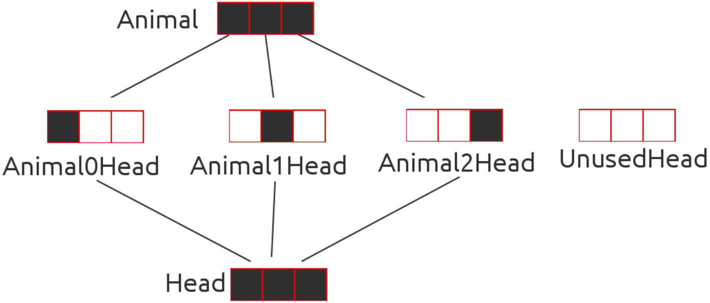
```
abstract Animal
    Head

Cat : Animal
Dog : Animal 2
```

$$scope(Cat) = 1, scope(Dog) = 2, scope(Animal) = 3, scope(Head) = 3$$

How do we join $SomeSetOfDog.Head$?

Abstract clafers

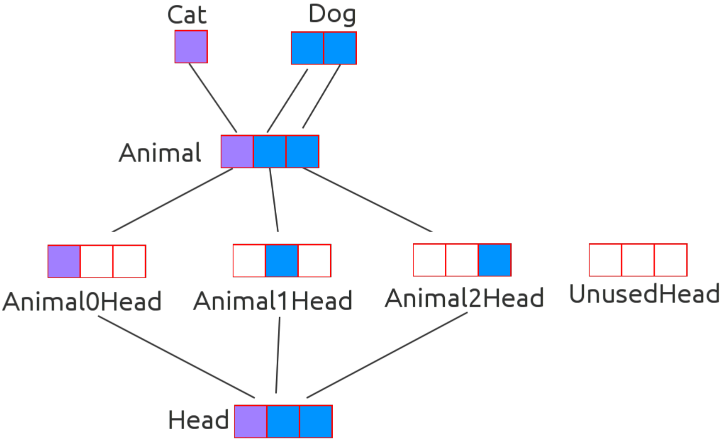
Abstract clafers have the same representation as concrete clafers.



Inherited clafers

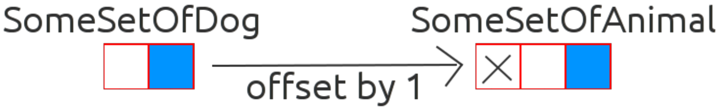
Dog and Cat "reserve" space in Animal.

$$Cat_0 = Animal_0, Dog_0 = Animal_1, Dog_1 = Animal_2$$



Upcasting

How do we join $SomeSetOfDog.Head$?
Suppose $SomeSetOfDog = \{Dog_1\}$.
First "upcast" $SomeSetOfDog$ to $SomeSetOfAnimal$.



Therefore $SomeSetOfAnimal = \{Animal_2\}$.
Then join $SomeSetOfAnimal.Head$ like previously described.

Working with set variables

For example, how do we implement the `offset` operation from the previous slide?
Let X be an experssion.

$$X \subseteq \{m, m+1, ..., n\}$$

Suppose we want to offset X by z. Create a new set variable Y.

$$Y \subseteq \{m+z, m+z+1, ..., n+z\} \text{ where } |Y| = |X|$$

Working with set variables

For example, how do we implement the `offset` operation from the previous slide?
Let X be an experssion.

$$X \subseteq \{m, m+1, ..., n\}$$

Suppose we want to offset X by z. Create a new set variable Y.

$$Y \subseteq \{m+z, m+z+1, ..., n+z\} \text{ where } |Y| = |X|$$

If $n - m$ is small

$$\bigwedge_{i=m}^{n} (i \in X \implies i + z \in Y)$$

Working with set variables

For example, how do we implement the `offset` operation from the previous slide?
Let X be an expersion.

$$X \subseteq \{m, m+1, ..., n\}$$

Suppose we want to offset X by z. Create a new set variable Y.

$$Y \subseteq \{m+z, m+z+1, ..., n+z\} \text{ where } |Y| = |X|$$

If $n - m$ is small

$$\bigwedge_{i=m}^{n} (i \in X \implies i + z \in Y)$$

If $b - a$ is small where $a \leq |X| \leq b$

Create new integer variables $y_0, y_1, ..., y_{b-a} \in \mathbb{Z}_{m,n}$.

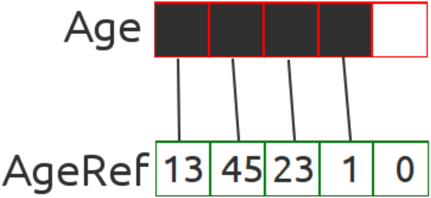
$$\bigwedge_{i=a}^{b} (|X| = i \implies allDifferent(y_0, ..., y_{i-1}) \wedge (\bigwedge_{j=0}^{i} y_i \in X \wedge y_i + z \in Y$$

References

Clafer 4: Age

```
Animal 4
    Age → integer
```

$$scope(Age) = 5$$



Clafer 5: Age solution

```
Animal0
    Age0 = 13
Animal1
    Age1 = 45
Animal2
    Age2 = 23
Animal3
    Age3 = 1
```

Optimize fixed size expressions

What if we knew the exact size of an expression at compile time? We can use an array of integer variables rather than a set variable.

$$
\begin{aligned}
SomeAgeSet &= [x] \\
SomeAgeSet.Ref &= [AgeRef[x]]
\end{aligned}
$$

In general, every Clafer expression has two implementations: one for variable size sets and the another for fixed size sets.

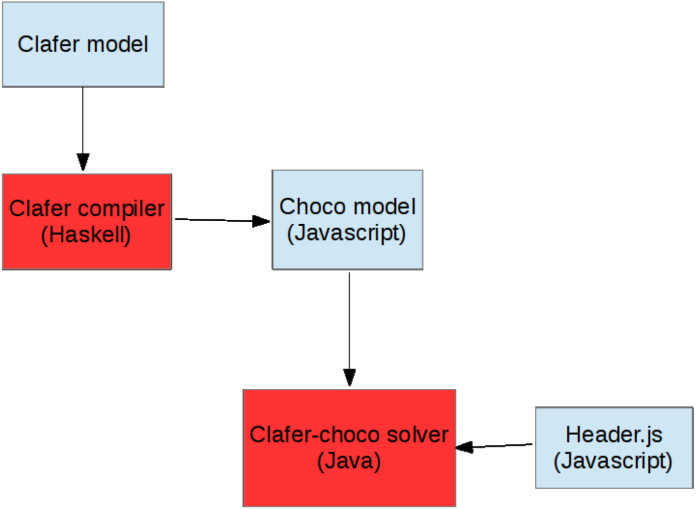
Reals/Strings?

- Choco only provides set variables for integers
- Fixed size expressions avoid sets
- Real and string support only viable for fixed size expressions (not yet implemented)
- Very restrictive

Clafer 6: Boat race starts after bike race

```
BikeRace 2
    EndTime → Real
BoatRace 2
    StartTime → Real

[BikeRace.EndTime = BoatRace.StartTime]
```

Implementation


Clafer model

Clafer compiler (Haskell)

Choco model (Javascript)

Clafer-choco solver (Java)

Header.js (Javascript)

Simple model test

Clafer 7: Animal kingdom

```
abstract Animal
    Head
        Eye 2
        Ear 2
        Mouth
    Age → integer
    Torso
    Leg 4
        Feet

Cat : Animal 4
    Whiskers 6
Rhino : Animal 3
    Horn
Elephant : Animal 2
    Trunk
```

Simple model test

- Alloy 4.2 with Minisat
- Default settings except extended heap to 2GB
- 6-year-old Dell laptop

Solver	Time to compute first solution
Alloy	15 min
Choco	2 sec

Realistic model test

- Linked list SPL, 18 features
- Model by Rafael Olaechea adapted from Scalable Prediction of Non-functional Properties in Software Product Lines
- Measurements are scaled down by a factor of 10

Solver	Time to compute solution			
	first (down)	first	optimum (down)	optimum
Alloy	11.5 sec	OOM (13 mins)	N/A	N/A
Choco	1.3 sec	1.3 sec	5.2 sec	5.3 sec

Future work

- Real and string support
- Custom constraints
  - Join with references
  - Offset
  - Sum of a set
  - Array of integer variables to set variable