Alcino Cunha

SPECIFICATION AND MODELING

RELATIONAL MODEL FINDING

Universidade do Minho & INESC TEC

2019/20

ARCHITECTURE

KODKOD

KODKOD

- Kodkod is a relational model finder
- A Kodkod *problem* consists of
  - a universe declaration: a set of atoms $\mathcal{U}$
  - relation declarations: for each relation r an arity $\mathrm{ar}(r)$ and lower and upper bounds (r_L and r_U)
  - a relational logic formula ϕ
- Kodkod finds values (a *model*) for the relations satisfying the given bounds and ϕ

FROM ALLOY TO KODKOD AND BACK

- Facts are added to the problem formula
- Formulas to be checked are negated and added to the problem formula
  - The formula is valid if its negation is unsatisfiable
- The main issue is how to infer (tight) bounds from scopes
  - Only atomic signatures and fields are declared
    - Non atomic signatures are aliased to disjunction of atomic ones
  - Upper bounds can be shared between disjoint atomic signatures
    - Further constraints must be added to ensure structural semantics
    - Atom names become meaningless
    - When building Alloy instance from Kodkod instance atoms must be renamed

ALLOY EXAMPLE

```
abstract sig HTTPEvent {}
sig Request extends HTTPEvent {
    response : lone Response
}
sig Response extends HTTPEvent {}
sig Redirect extends Response {}
run {response.Redirect in HTTPEvent} for 3 but exactly 1 Redirect
```

KODKOD TRANSLATION

```
{A,B,C}

Request   :1 {} {(A),(B)}
$Response :1 {} {(A),(B)}
Redirect  :1 {(C)} {(C)}
response  :2 {} {(A,A),(A,B),(A,C),(B,A),(B,B),(B,C)}

no Request & ($Response+Redirect)
all x : Request | lone x.response and x.response in ($Response+Redirect)
response.univ in Request
response.Redirect in (Request+$Response+Redirect)
```

SAT

BOOLEAN SATISFIABILITY

- The boolean satisfiability problem (SAT) is the problem of determining if a propositional logic formula has a (valid) model
- SAT was the first problem to be proven NP-complete
- But modern SAT solvers can handle formulas with tens of thousands of variables efficiently

FROM KODKOD TO SAT AND BACK

- A relation r of arity $k = \mathrm{ar}(r)$ can be represented by a k-dimensional matrix $|\mathcal{U}|^k$ of propositional variables

$$r[i_1, \ldots, i_k] = \begin{cases} \top & \text{if } \langle \mathcal{U}_{i_1}, \ldots, \mathcal{U}_{i_k} \rangle \in r_L \\ x_{i_1,\ldots,i_k} & \text{if } \langle \mathcal{U}_{i_1}, \ldots, \mathcal{U}_{i_k} \rangle \in r_U \setminus r_L \\ \bot & \text{otherwise} \end{cases}$$

- Relational operators implemented by matrix operations and boolean connectives
  - Composition is the product
  - Union is the sum
  - Intersection is Hadamard product
  - Closure iterates $|\mathcal{U}|$ products
  - Inclusion is implication
  - …

KODKOD EXAMPLE

```
{A,B,C}

Request   :1 {} {(A),(B)}
$Response :1 {} {(A),(B)}
Redirect  :1 {(C)} {(C)}
response  :2 {} {(A,A),(A,B),(A,C),(B,A),(B,B),(B,C)}

no Request & ($Response+Redirect)
all x : Request | lone x.response and x.response in ($Response+Redirect)
response.univ in Request
response.Redirect in (Request+$Response+Redirect)
```

SAT TRANSLATION

```
response.Redirect in (Request+$Response+Redirect)
```

$$\begin{bmatrix} r_{A,A} & r_{A,B} & r_{A,C} \\ r_{B,A} & r_{B,B} & r_{B,C} \\ \bot & \bot & \bot \end{bmatrix} \cdot \begin{bmatrix} \bot \\ \bot \\ \top \end{bmatrix} \subseteq \begin{bmatrix} x_A \\ x_B \\ \bot \end{bmatrix} + \begin{bmatrix} y_A \\ y_B \\ \bot \end{bmatrix} + \begin{bmatrix} \bot \\ \bot \\ \top \end{bmatrix}$$

$$\begin{bmatrix} r_{A,A} \wedge \bot \ \vee \ r_{A,B} \wedge \bot \ \vee \ r_{A,C} \wedge \top \\ r_{B,A} \wedge \bot \ \vee \ r_{B,B} \wedge \bot \ \vee \ r_{B,C} \wedge \top \\ \bot \wedge \bot \ \vee \ \bot \wedge \bot \ \vee \ \bot \wedge \top \end{bmatrix} \subseteq \begin{bmatrix} x_A \vee y_A \vee \bot \\ x_B \vee y_B \vee \bot \\ \bot \vee \bot \vee \top \end{bmatrix}$$

$$(r_{A,C} \rightarrow x_A \vee y_A) \wedge (r_{B,C} \rightarrow x_B \vee y_B) \wedge (\bot \rightarrow \top)$$

QUANTIFIERS

- Since the universe is finite, quantifiers can be handled by expansion

```
all x : Request | lone x.response
```

≡

```
lone {(A)}.response and lone {(B)}.response
```

- Unfortunately, this technique yields no witnesses to existential quantifiers

```
some x : Request | some x.response
```

≡

```
some {(A)}.response or some {(B)}.response
```

SKOLEMIZATION

- *Skolemization* is a technique that replaces existentially quantified variables by new free variables.
    - Free variables are implicitly existentially quantified
    - Generates smaller but equisatisfiable formulas

```
some x : Request | some x.response
```

≅

```
$x :1 {} {(A),(B)}
one $x and some $x.response
```

- Skolemized variables are witnesses of the quantifier and are very useful in visualisation
- Skolemization can also be applied to higher-order existential quantifications

SYMMETRY BREAKING

- Kodkod performs several optimisations to decrease SAT complexity
- The most significant is *symmetry breaking*
  - Since atoms are uninterpreted (almost) any permutation of an instance is also a valid instance
  - A symmetry-breaking formula is conjoined to the problem formula
  - It tries to capture most symmetries but for efficiency reasons the technique is not complete
- Besides improving efficiency, symmetry breaking is also great for validation
  - Without it the user would be overwhelmed with isomorphic instances