How is Model-Checking Done?

- Semantics of propositions
- Tableau-based checking:
  - Notion of tableau
  - Rules
  - Example
- Implementation of model-checking
- Running times

Terminology

$\mathcal{A}$ - a set of atomic formulas $\mathcal{A}$.
$\forall$ (disjoint from $\mathcal{A}$) - a set of propositional variables $X$.
$Act$ - a set of actions $a$.
Formulas are $\Phi$...
$T$ - a set of states

- $\Gamma \prec \Phi$ if $\Gamma$ is a strict subformula of $\Phi$.
- environment - a mapping of variables to sets of states as a means of interpreting free propositional variables.
- $e[X \mapsto S]$ - the environment $e$ with $X$ "updated" to $S$.
- Use sequents of the form $H \vdash s \in \Phi$, where $s$ is a state, $\Phi$ is a formula, and $H$ is a set of hypotheses of the form $\forall \cdot \Gamma$, for $s$ a state and $\Gamma$ a closed recursive formula.
Semantics of propositions

\[ [A]e = V(A) \]
\[ [X]e = e(X) \]
\[ [-\Phi]e = \top \quad [\Phi]e \]
\[ [\Phi_1 \lor \Phi_2]e = [\Phi_1]e \cup [\Phi_2]e \]
\[ [<a>\Phi]e = \pi_a([\Phi]e), \text{ where} \]
\[ \pi_a(S) = \{ s' \mid \exists s \in S, s \approx a s' \} \]
\[ [\nu X.\Phi]e = \bigcup \{ S \subseteq \mathcal{T} \mid S \subseteq [\Phi]e[X \mapsto S] \} \]

Idea of tableau

- Theorem: \( H \vdash s \in \Phi \) has a successful tableau if and only if \( H \vdash s \in -\Phi \) has no successful tableau.
- Idea: start with property (or negated property), apply rules R1-R8 and DR1-DR3 (below) in top-down fashion until all leaves are successful. A leaf is successful if and only if one of the following holds:

1. \( \Phi \in \mathcal{A} \) and \( s \in V(\Phi) \).
2. \( \Phi \) is \(-A\) for some \( A \in \mathcal{A} \) and \( s \notin V(A) \).
3. \( \Phi \) is \(-<a>\Phi\) for some \( a \) and \( \Phi\).
4. \( \Phi \) is \( \nu X.\Phi\) for some \( X \) and \( \Phi\).
5. Sequents of form \( H \vdash s \in T_{true} \) are successful.
6. Leaves of the form \( H \vdash s \in [a]\Phi \) are successful.

Note: \( H \vdash s \in -<a>\Phi \) is a leaf only when \( s \) has no \( a \)-derivatives, while \( H \vdash s \in \nu X.\Phi \) is a leaf only when \( s : \nu X.\Phi \in H \).
Rules

see Figure 3 on p. 730 of Acta Informatica paper

Rules (Cont’d)

see Figure 4 on p. 732 of Acta Informatica paper
Rules, Etc.

R7 and R8 require that in order to establish that a state enjoys a (negated) recursive property, it is sufficient to establish that it enjoys the (negated) unrolling of the property, provided that the assumptions involving the formulas having the recursive formula as a subformula are removed or discharged from the hypothesis list.

Other results:
1. (Finiteness) If models are finite, their tableaux are finite.
2. (Soundness and completeness) \( H \vdash s \in \Phi \) has a successful tableau if and only if \( s \in \llbracket \Phi \rrbracket^H \).

Example

See Figure 5 on p. 732 of Acta Informatica paper
Simple Implementation of model-checking

fun check1v(H ⊨ s ∈ Φ) =
    case Φ is
    A ∈ A → return (s ∈ V(A))
    X ∈ V → error
    ¬Φ1 → return not (check1v(H ⊨ s ∈ Φ1))
    Φ1 ∨ Φ2 → return (check1v(H ⊨ s ∈ Φ1)
        or check1v(H ⊨ s ∈ Φ2))
    < a > Φ1 → for each s1 ∈ {s2 ⊢ s1} do
        if check1v(H ⊨ s1 ∈ Φ1) then
            return true;
        else return false
    νX.Φ1 → let H1 = {s1 : Γ | Φ1 ⊨ s1} in
        return (check1v(H1 ⊨ s : Φ1 ⊨ s ∈ Φ1[Φ/X]))
end
fun check1(s ∈ Φ) = check1v(∅ ⊨ s ∈ Φ)

Running times

- Algorithm has exponential running time even for formulas having no recursive subformulas, owing to the possibility of nested modal operators.
- Possible optimization: store results of sequents whose truth has already been determined
- Running time is $O((|S| \times |Φ|)^{id(Φ)+1})$:
  - $id(Φ)$ = interconnection depth of $Φ$, measure of the degree of mutual recursion in $Φ$
  - $Φ$ = formula under verification
  - $S$ = number of states in transition system