Model Checking is Static Analysis

Flemming Nielson, DTU Informatics

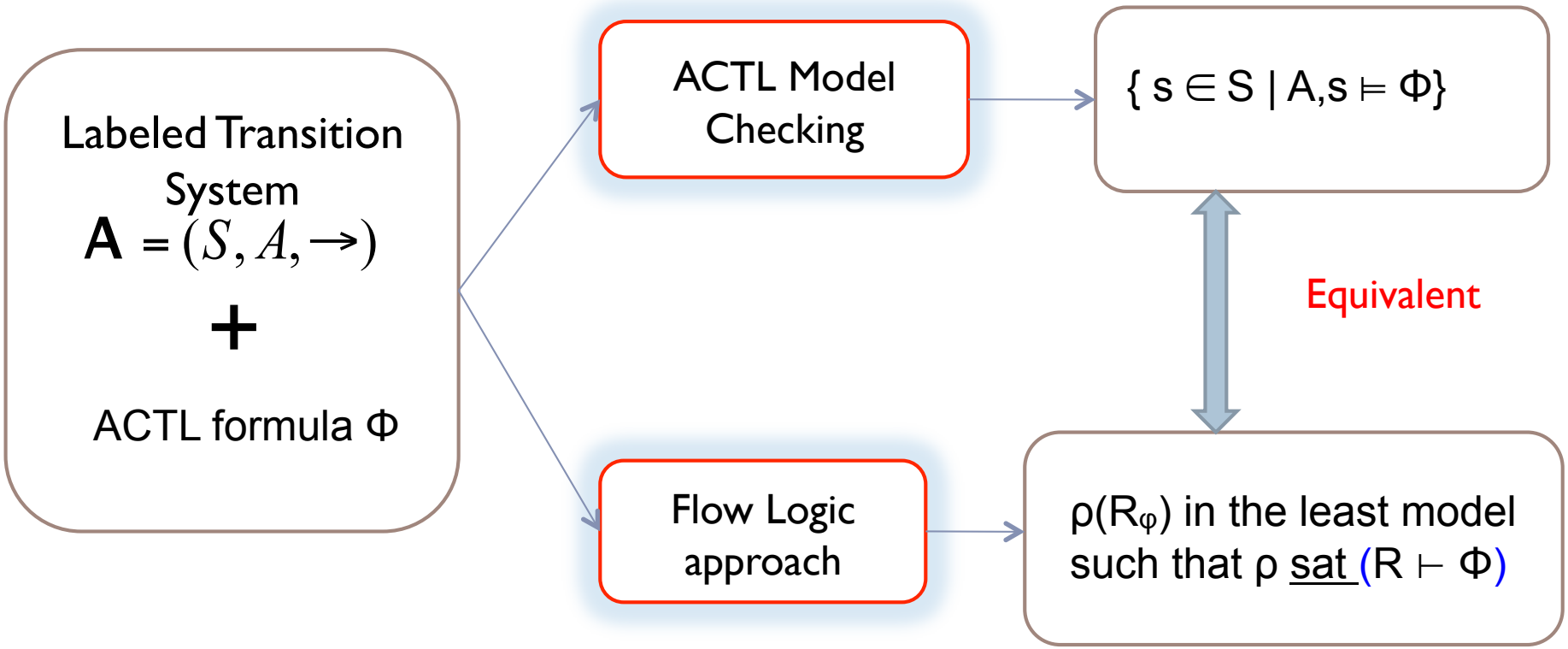
Problems versus Algorithms

- How are the problems of Model Checking and Static Analysis related to each other?
  - From Static Analysis to Model Checking: the previous talk
  - From Model Checking to Static Analysis: this talk
- How are the iterative algorithms of Model Checking and Static Analysis related to each other?
- How are the abstraction techniques of Model Checking and Static Analysis related to each other?

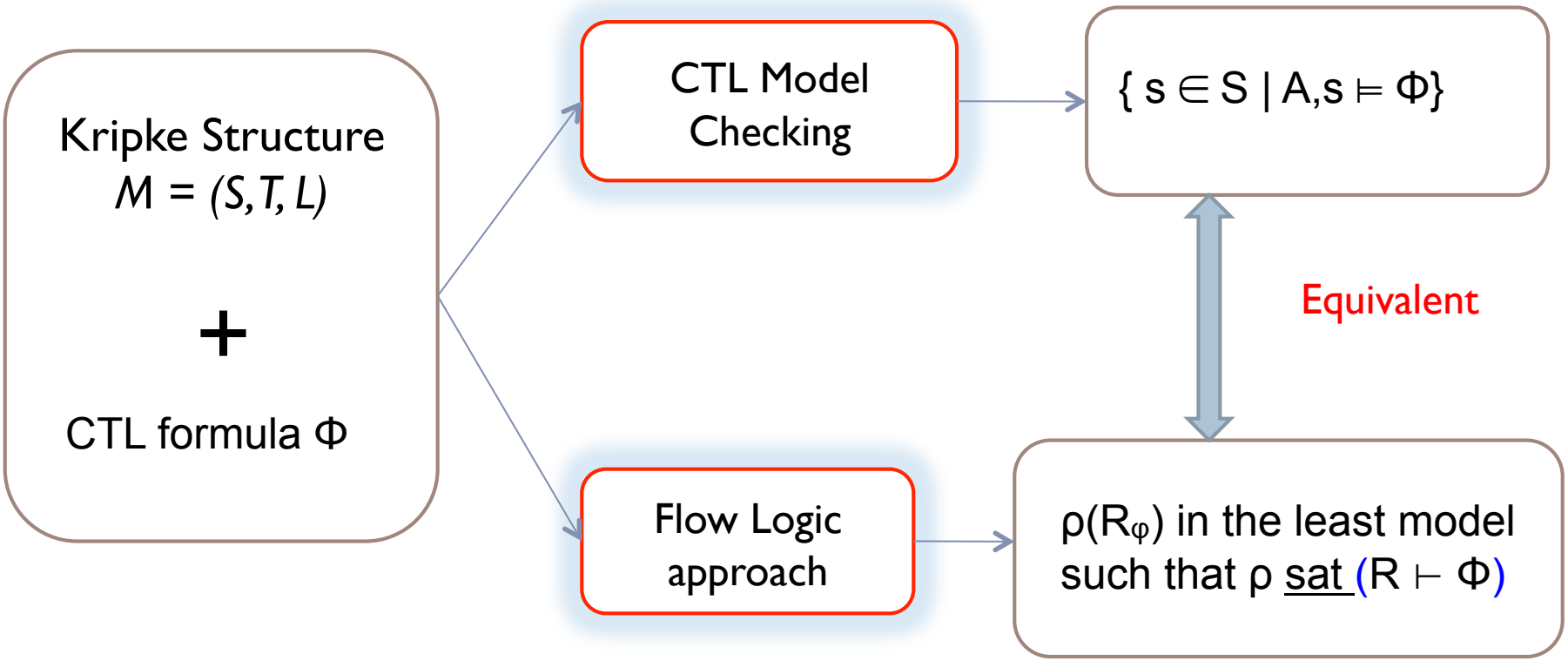
Current Results

Model Checking is Static Analysis

ACTL Model Checking is Static Analysis

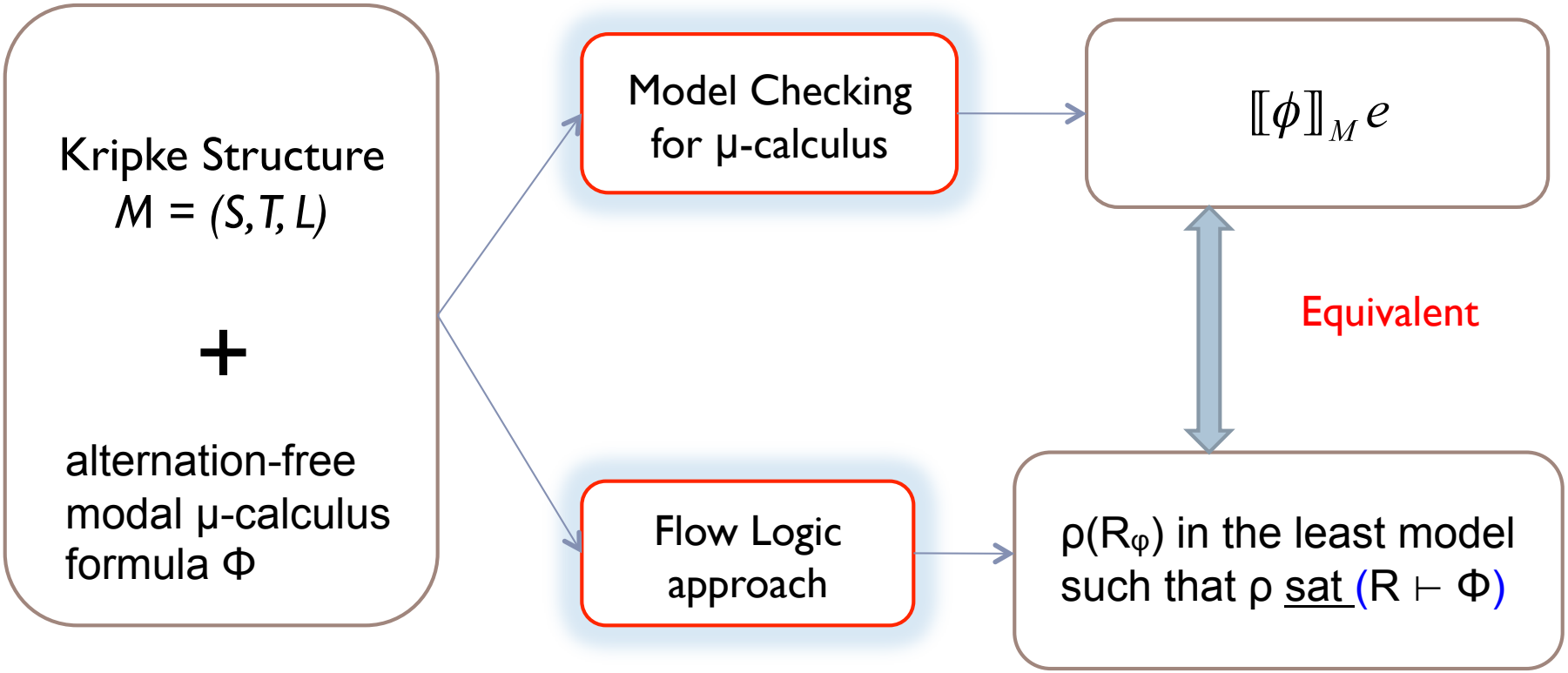


CTL Model Checking is Static Analysis



Flemming Nielson, Hanne Riis Nielson, Piotr Filipiuk, Fuyuan Zhang @ unpublished

Alternation-free μ-calculus Model Checking is Static Analysis



Fuyuan Zhang, Flemming Nielson @ unpublished

Model Checking for ACTL

Model Checking is Static Analysis

Labelled Transition Systems

A *labelled transition system* (LTS) has the form $(S, \mathcal{A}, \rightarrow)$ where

- S is a non-empty set of states,
- $\mathcal{A}$ is a non-empty set of actions and
- $\rightarrow \subseteq S \times \mathcal{A} \times S$ is the transition relation.

We write $s \rightarrow^a s'$ whenever $(s, a, s') \in \rightarrow$.

A *path* π is a *maximal* sequence $(s_i \rightarrow^{a_i} s_{i+1})_{0 \leq i < n}$ such that $s_i \rightarrow^{a_i} s_{i+1}$ for all $i \geq 0$ and where $n \in \{0, \cdots, \infty\}$ is the length of the path.

Action Computation Tree Logic

We shall use a variant of the modal logic *Action Computation Tree Logic* (ACTL) to express properties of paths in labelled transition systems:

$$
\begin{array}{rcll}
\phi & ::= & \text{true} \mid \text{false} \mid bp & \\
 & \mid & \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg\phi \mid \phi_1 \Rightarrow \phi_2 & \\
 & \mid & \mathbf{EX}_\Omega\ \phi \mid \mathbf{AX}_\Omega\ \phi & (\Omega \subseteq \mathcal{A}) \\
 & \mid & \mathbf{E}[\phi_1\ {}_{\Omega_1}\mathbf{U}_{\Omega_2}\ \phi_2] \mid \mathbf{A}[\phi_1\ {}_{\Omega_1}\mathbf{U}_{\Omega_2}\ \phi_2] & (\Omega_i \subseteq \mathcal{A}) \\
bp & ::= & \cdots & \text{(basic predicates)}
\end{array}
$$

- **EX** $\cdot$: exists next step
- **AX** $\cdot$: forall next steps
- **E**[$\cdot$ **U** $\cdot$]: exists path until
- **A**[$\cdot$ **U** $\cdot$]: forall paths until

Interpretation of ACTL

The interpretation $s \models \phi$ is defined relative to the state s:

$$
\begin{aligned}
s &\models \text{true} && \underline{\text{iff}} && \text{true} \\
s &\models \text{false} && \underline{\text{iff}} && \text{false} \\
s &\models bp && \underline{\text{iff}} && bp \text{ holds in } s \\
s &\models \phi_1 \wedge \phi_2 && \underline{\text{iff}} && (s \models \phi_1) \wedge (s \models \phi_2) \\
s &\models \phi_1 \vee \phi_2 && \underline{\text{iff}} && (s \models \phi_1) \vee (s \models \phi_2) \\
s &\models \neg\phi && \underline{\text{iff}} && \neg (s \models \phi) \\
s &\models \phi_1 \Rightarrow \phi_2 && \underline{\text{iff}} && (s \models \phi_1) \Rightarrow (s \models \phi_2)
\end{aligned}
$$

Interpretation of ACTL

$$s_0 \models \mathbf{EX}_\Omega\ \phi \quad \underline{\text{iff}} \quad \exists (s_i \rightarrow^{a_i} s_{i+1})_{0 \leq i < n} : n > 0 \wedge a_0 \in \Omega \wedge s_1 \models \phi$$

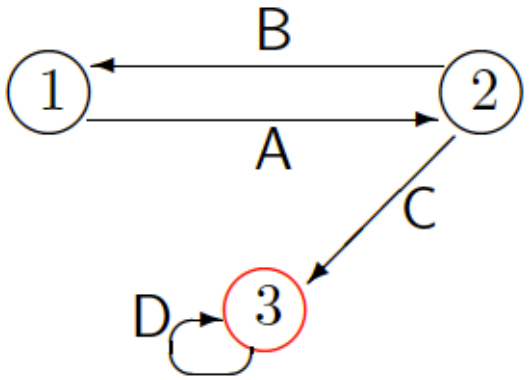
$$s_0 \models \mathbf{AX}_\Omega\ \phi \quad \underline{\text{iff}} \quad \forall (s_i \rightarrow^{a_i} s_{i+1})_{0 \leq i < n} : n > 0 \wedge a_0 \in \Omega \wedge s_1 \models \phi$$

$$s_0 \models \mathbf{E}[\phi_1\ {}_{\Omega_1}\mathbf{U}_{\Omega_2}\ \phi_2] \quad \underline{\text{iff}} \quad \exists (s_i \rightarrow^{a_i} s_{i+1})_{0 \leq i < n} : \exists k < n : [\bigwedge_{0 \leq i < k} (a_i \in \Omega_1 \wedge s_{i+1} \models \phi_1) \wedge (a_k \in \Omega_2 \wedge s_{k+1} \models \phi_2)]$$

$$s_0 \models \mathbf{A}[\phi_1\ {}_{\Omega_1}\mathbf{U}_{\Omega_2}\ \phi_2] \quad \underline{\text{iff}} \quad \forall (s_i \rightarrow^{a_i} s_{i+1})_{0 \leq i < n} : \exists k < n : [\bigwedge_{0 \leq i < k} (a_i \in \Omega_1 \wedge s_{i+1} \models \phi_1) \wedge (a_k \in \Omega_2 \wedge s_{k+1} \models \phi_2)]$$

An Example

Transition system:



Let "goal" be the predicate that is only satisfied in the state 3.

ϕ	$\{s \mid s \models \phi\}$
$\mathbf{EX}_{\mathcal{A}}$ goal	$\{2,3\}$
$\mathbf{AX}_{\mathcal{A}}$ goal	$\{3\}$
$\mathbf{E}[\text{true } {}_{\mathcal{A}}\mathbf{U}_{\mathcal{A}} \text{ goal}]$	$\{1,2,3\}$
$\mathbf{A}[\text{true } {}_{\mathcal{A}}\mathbf{U}_{\mathcal{A}} \text{ goal}]$	$\{3\}$
$\mathbf{AG}_{\mathcal{A}}$ goal	$\{3\}$
$\mathbf{AG}_{\{C\}}$ goal	$\{1,2,3\}$

Static Analysis using Flow Logic

Model Checking is Static Analysis

Flow Logic

Flow Logic bridges the gap between a number of approaches to static analysis including Data Flow Analysis, Constraint Based Analysis, Abstract Interpretation and Type and Effect Systems.

In Flow Logic *logical judgements* are used to *specify* when the analysis information correctly captures the information of the program or system.

The *correctness* of the analysis is established as a subject reduction result; we usually do not have precision.

To ensure that the analysis is *implementable* it is customary to establish a Moore family, or model intersection property.

The Succinct Solver often suffices for producing polynomial time *implementations*.

Alternation-free Least Fixed Point Logic

The Moore family result guarantees the existence of best analysis results but in itself it does not provide any mechanism for constructing the analysis result.

Alternation-free Least Fixed Point Logic is a generalisation of Datalog:

$$
\begin{array}{lll}
\text{values:} & v & ::= \quad c \mid x \mid f(v_1, \ldots, v_k) \\
\text{preconditions:} & pre & ::= \quad R(v_1, \ldots, v_k) \mid \neg R(v_1, \ldots, v_k) \\
 & & \quad \mid \quad pre_1 \wedge pre_2 \mid pre_1 \vee pre_2 \\
 & & \quad \mid \quad \forall x : pre \mid \exists x : pre \\
\text{clauses:} & cl & ::= \quad R(v_1, \ldots, v_k) \mid \text{true} \mid cl_1 \wedge cl_2 \\
 & & \quad \mid \quad pre \Rightarrow cl \mid \forall x : cl
\end{array}
$$

Interpretation of ALFP

Satisfaction (ϱ, σ) sat cl is defined relative to a universe $\mathcal{U}$; here ϱ an interpretation of relations and σ is an interpretation of variables.

For preconditions:

$$
\begin{array}{llll}
(\varrho,\sigma) & \underline{\text{sat}} & R(v_1,\ldots,v_k) & \underline{\text{iff}} \quad (\sigma(v_1),\ldots,\sigma(v_k)) \in \varrho(R) \\
(\varrho,\sigma) & \underline{\text{sat}} & \neg R(v_1,\ldots,v_k) & \underline{\text{iff}} \quad (\sigma(v_1),\ldots,\sigma(v_k)) \notin \varrho(R) \\
(\varrho,\sigma) & \underline{\text{sat}} & pre_1 \wedge pre_2 & \underline{\text{iff}} \quad (\varrho,\sigma) \ \underline{\text{sat}} \ pre_1 \text{ and } (\varrho,\sigma) \ \underline{\text{sat}} \ pre_2 \\
(\varrho,\sigma) & \underline{\text{sat}} & pre_1 \vee pre_2 & \underline{\text{iff}} \quad (\varrho,\sigma) \ \underline{\text{sat}} \ pre_1 \text{ or } (\varrho,\sigma) \ \underline{\text{sat}} \ pre_2 \\
(\varrho,\sigma) & \underline{\text{sat}} & \forall x : pre & \underline{\text{iff}} \quad (\varrho,\sigma[x \mapsto a]) \ \underline{\text{sat}} \ pre \\
 & & & \qquad \text{for all } a \in \mathcal{U} \\
(\varrho,\sigma) & \underline{\text{sat}} & \exists x : pre & \underline{\text{iff}} \quad (\varrho,\sigma[x \mapsto a]) \ \underline{\text{sat}} \ pre \\
 & & & \qquad \text{for some } a \in \mathcal{U}
\end{array}
$$

Interpretation of ALFP

For clauses:

$$
\begin{array}{lll}
(\varrho,\sigma) \ \underline{\text{sat}} \ R(v_1,\ldots,v_k) & \underline{\text{iff}} & (\sigma(v_1),\ldots,\sigma(v_k)) \in \varrho(R) \\
(\varrho,\sigma) \ \underline{\text{sat}} \ \text{true} & \underline{\text{iff}} & \text{true} \\
(\varrho,\sigma) \ \underline{\text{sat}} \ cl_1 \wedge cl_2 & \underline{\text{iff}} & (\varrho,\sigma) \ \underline{\text{sat}} \ cl_1 \text{ and } (\varrho,\sigma) \ \underline{\text{sat}} \ cl_2 \\
(\varrho,\sigma) \ \underline{\text{sat}} \ pre \Rightarrow cl & \underline{\text{iff}} & (\varrho,\sigma) \ \underline{\text{sat}} \ cl \\
 & & \quad \text{whenever } (\varrho,\sigma) \ \underline{\text{sat}} \ pre \\
(\varrho,\sigma) \ \underline{\text{sat}} \ \forall x : cl & \underline{\text{iff}} & (\varrho,\sigma[x \mapsto a]) \ \underline{\text{sat}} \ cl \\
 & & \quad \text{for all } a \in \mathcal{U}
\end{array}
$$

Stratification

A clause cl is *stratified* if there is a number r, an assignment of numbers called ranks $\text{rank}_R \in \{0, \cdots, r\}$ to each relation R, and a way to write cl on the form $\bigwedge_{0 \leq i \leq r} cl_i$ such that the following holds for all clauses:

- if cl_i contains a definition of R then $\text{rank}_R \geq i$;
- if cl_i contains a positive use of R then $\text{rank}_R \leq i$; and
- if cl_i contains a negative use of R then $\text{rank}_R < i$.

Stratification induces a lexicographic order that is also a partial order.

Theorem [Nielson,Nielson,Seidl]

The set $\{\varrho \mid (\varrho, \sigma_0)\ \underline{\text{sat}}\ cl\}$ is a Moore family whenever cl is closed and stratified;
the greatest lower bound $\sqcap \{\varrho \mid (\varrho, \sigma_0)\ \underline{\text{sat}}\ cl\}$ is the *least* model of cl.

The Encoding

Model Checking is Static Analysis

Encoding ACTL in ALFP

For each formula ϕ of ACTL we define a relation R_ϕ (of ALFP) containing those states where the formula ϕ holds (and perhaps more); the analysis judgements $\vec{R} \vdash \phi$ (in ALFP) defines the relation.

Idea:

$s \models \phi$ holds (for ACTL) whenever $R_\phi(s)$ holds in the *least model* satisfying $\vec{R} \vdash \phi$ (for ALFP).

We assume:

- for each basic predicate bp we have a relation P_{bp} on states,
- for each subset Ω of $\mathcal{A}$ we have a relation Ω on actions, and
- the transition system is presented by a ternary relation T.

The non-modal operators

$$
\begin{array}{rcl}
\vec{R} \vdash \mathsf{true}^{\ell} & \underline{\text{iff}} & [\forall s : R_{\mathsf{true}^{\ell}}(s)] \\
\vec{R} \vdash \mathsf{false}^{\ell} & \underline{\text{iff}} & \mathsf{true} \\
\vec{R} \vdash bp^{\ell} & \underline{\text{iff}} & [\forall s : P_{bp}(s) \Rightarrow R_{bp^{\ell}}(s)] \\
\vec{R} \vdash (\phi_1^{\ell_1} \vee \phi_2^{\ell_2})^{\ell} & \underline{\text{iff}} & \vec{R} \vdash \phi_1^{\ell_1} \wedge \vec{R} \vdash \phi_2^{\ell_2} \wedge \\
 & & [\forall s : R_{\phi_1^{\ell_1}}(s) \vee R_{\phi_2^{\ell_2}}(s) \Rightarrow R_{(\phi_1^{\ell_1} \vee \phi_2^{\ell_2})^{\ell}}(s)] \\
\vec{R} \vdash (\neg \phi^{\ell'})^{\ell} & \underline{\text{iff}} & \vec{R} \vdash \phi^{\ell'} \wedge \\
 & & [\forall s : (\neg\, R_{\phi^{\ell'}}(s)) \Rightarrow R_{(\neg \phi^{\ell'})^{\ell}}(s)] \\
\vec{R} \vdash (\phi_1^{\ell_1} \Rightarrow \phi_2^{\ell_2})^{\ell} & \underline{\text{iff}} & \vec{R} \vdash \phi_1^{\ell_1} \wedge \vec{R} \vdash \phi_2^{\ell_2} \wedge \\
 & & [\forall s : \neg\, R_{\phi_1^{\ell_1}}(s) \vee R_{\phi_2^{\ell_2}}(s) \Rightarrow R_{(\phi_1^{\ell_1} \Rightarrow \phi_2^{\ell_2})^{\ell}}(s)]
\end{array}
$$

The next modalities

$$\vec{R} \vdash (\mathbf{EX}_\Omega\, \phi^{\ell'})^{\ell} \quad \underline{\text{iff}} \quad \vec{R} \vdash \phi^{\ell'} \wedge [\forall s : [\exists a : \exists s' : T(s,a,s') \wedge \Omega(a) \wedge R_{\phi^{\ell'}}(s')] \Rightarrow R_{(\mathbf{EX}_\Omega\, \phi^{\ell'})^{\ell}}(s)]$$

$$\vec{R} \vdash (\mathbf{AX}_\Omega\, \phi^{\ell'})^{\ell} \quad \underline{\text{iff}} \quad \vec{R} \vdash \phi^{\ell'} \wedge [\forall s : [\forall a : \forall s' : \neg T(s,a,s') \vee (\Omega(a) \wedge R_{\phi^{\ell'}}(s'))] \wedge [\exists a : \exists s' : T(s,a,s')] \Rightarrow R_{(\mathbf{AX}_\Omega\, \phi^{\ell'})^{\ell}}(s)]$$

Note that we step outside the Datalog fragment and use all of stratified ALFP.

The until modalities

$$\vec{R} \vdash (\mathbf{A}[\phi_1^{\ell_1}\ \Omega_1 \mathbf{U}_{\Omega_2}\ \phi_2^{\ell_2}])^{\ell} \quad \underline{\text{iff}} \quad \vec{R} \vdash \phi_1^{\ell_1} \wedge \vec{R} \vdash \phi_2^{\ell_2} \wedge$$
$$[\forall s : [\ [\exists a : \exists s' : T(s,a,s')] \wedge$$
$$[\forall a : \forall s' : \ \neg T(s,a,s') \vee$$
$$[\Omega_2(a) \wedge R_{\phi_2^{\ell_2}}(s')] \vee$$
$$[\Omega_1(a) \wedge R_{\phi_1^{\ell_1}}(s') \wedge$$
$$R_{(\mathbf{A}[\phi_1^{\ell_1}\ \Omega_1 \mathbf{U}_{\Omega_2}\ \phi_2^{\ell_2}])^{\ell}}(s')]\ ]$$
$$\Rightarrow R_{(\mathbf{A}[\phi_1^{\ell_1}\ \Omega_1 \mathbf{U}_{\Omega_2}\ \phi_2^{\ell_2}])^{\ell}}(s)]$$

The formula for E is slightly simpler.

Achieving Stratification

We shall require that all sub-formula of an ACTL formula are annotated with their number in a *post-order traversal* of the formula.

To be specific we shall say that a formula ϕ is (i, j)-*annotated* for positive integers i and j if the lowest annotation occurring within ϕ is i and the highest is j – the annotation of the formula ϕ itself will then be j.

Theorem

The ALFP clauses generated above from an $(1, \ell)$-annotated ACTL formula ϕ is closed and stratified.

A closed and stratified (i, j)-annotated formula is called an (i, j)-*stratified* formula.

Correctness and Precision

Theorem

Consider an (i,j)-stratified formulae ϕ^j in ACTL and the least model ϱ of $\vec{R} \vdash \phi$ such that $\varrho \sqsupseteq \varrho_0$. We then have:

- Correctness of ϱ: if $s \models \phi^j$ then $\varrho(R_{\phi^j})(s)$.
- Precision of ϱ: if $\varrho(R_{\phi^j})(s)$ then $s \models \phi^j$.

Correctness is the usual result established for static analyses; precision only hold for analyses "close to the semantics" like the Collecting Semantics of Abstract Interpretation.

Corollary

The implementation of the static analysis by means of the Succinct Solver constitutes a *model checker* for ACTL!

The Computational Complexity

Model Checking is Static Analysis

Succinct Solver: Complexity

The *Succinct Solver* is a software tool that computes the least model guaranteed by the Moore family theorem for ALFP.

The worst case time complexity is given by:

Theorem [Nielson,Nielson,Seidl]

Under the assumptions of the previous theorem the least model $\sqcap \{\varrho \mid (\varrho, \sigma_0) \underline{\text{sat}}\ cl \wedge \varrho_0 \sqsubseteq \varrho\}$ is computable in time

$$\mathcal{O}(|\varrho_0| + |\mathcal{U}|^K |cl|)$$

where $|\varrho_0|$ is the size of ϱ_0 and $|\mathcal{U}|$ is the size of the (necessarily finite) universe $\mathcal{U}$ and K is the maximal nesting depth of quantifiers within cl.

Succinct Solver: Technology

The *Succinct Solver* often exhibits a running time substantially lower than the worst case time complexity.

- It is programmed in OCAML (or Standard ML).
- It deals with stratification by computing the relations in increasing order on their rank.
- It combines the top-down solving approach of Le Charlier and van Hentenryck with the propagation of differences (distributive frameworks, deductive databases, reduction of strength transformations).
- Disciplined use of continuations and memoisation as well as arbitrarily branching prefix trees as a universal data-structure for storing relations and for organising sets of waiting consumers.

Complexity of Model Checking

Consider a transition system $(S, \mathcal{A}, \rightarrow)$ where the state space S has size $|S|$ and the transition relation $\rightarrow$ has size $|T|$ and consider an ACTL formula ϕ of size $|\phi|$.

The ALFP clause $\vec{R} \vdash \phi$ has size $\mathcal{O}(|\phi|)$ and nesting depth 3 and the worst case time complexity is

$$\mathcal{O}(|T| + \sum_{bp} |P_{bp}| + 2^{|\mathcal{A}|} + |S|^3 |\phi|)$$

When the number of base predicates and the number of actions is bounded by some constant then a more refined reasoning gives

$$\mathcal{O}((|T| + |S|)\,|\phi|)$$

This equals the worst case complexity for model checking ACTL.

MT-LAB
A VKR CENTRE OF EXCELLENCE

The Conclusion

Model Checking is Static Analysis

The Conclusion

- How are the problems of Model Checking and Static Analysis related to each other?
  - From Model Checking to Static Analysis:
    - CTL-type logics can be encoded
    - Alternation-free modal μ–calculus can be encoded
    - The borderline is not yet clear – full modal μ–calculus may invalidate the Moore Family approach ?
  - From Static Analysis to Model Checking:
    - Some static analysis problems can be encoded – but which ones?

The Conclusion

- How are the iterative algorithms of Model Checking and Static Analysis related to each other?
- How are the abstraction techniques of Model Checking and Static Analysis related to each other?
- The thesis of Theme 1 of MT-LAB (a VKR Centre of Excellence www.MT-LAB.dk) is:
  - Static analysis and model checking fundamentally solve the same problem – but using a different repertoire of techniques that must be combined in order to produce more powerful analysis techniques.