Descriptive Complexity and Model Checking

Neil Immerman

Computer Science Dept.
UMass. Amherst

http://www.cs.umass.edu/∼immerman

Model Checking

Emerson, Clarke, ...

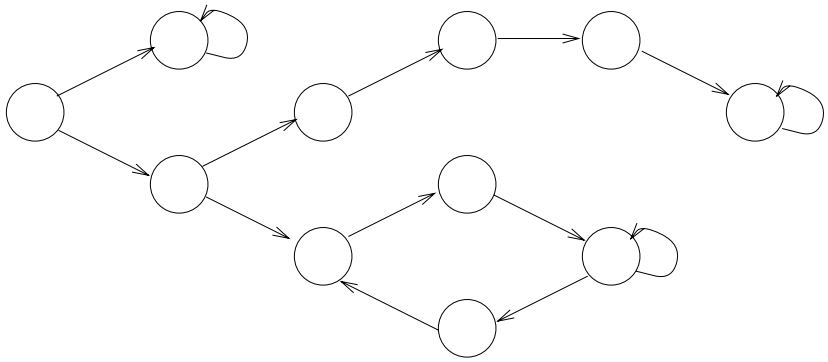
1. Encode a program, protocol, or circuit in a formal language, e.g., JAVA or VHDL.

2. Write a correctness assertion in a formal language, e.g., CTL* or FO(TC), e.g.,
   (a) If the Restart button is pressed, we eventually restart.
   (b) Doors are not opened between stations.
   (c) Division is performed correctly.
   (d) The motor is not turned on during maintenance.

3. Press a button and find out quickly whether or not your design satisfies the assertion.

Temporal Logic: LTL , CTL $\subset$ CTL*

Kripke structure: $\mathcal{K} = (S, R, p_1, \ldots, p_t)$



Examples of Temporal Logic:

$$\begin{aligned}
\mathbf{EF}p &\equiv \text{“We can get to a state where } p \text{ holds”} \\
\mathbf{AG}p &\equiv \text{“}p \text{ holds at all reachable states”} \\
\mathbf{EG}p &\equiv \text{“Along some path } p \text{ always holds”}
\end{aligned}$$

$$\mathbf{AG}(p \rightarrow \mathbf{EX}q) \equiv \text{“If } p\text{, then at some next state, } q\text{.”}$$

$$\begin{aligned}
\mathbf{EGF}p &\equiv \text{“Along some path } p \text{ holds infinitely often”} \\
\mathbf{AG}(\mathbf{G}\ \text{request} \rightarrow \mathbf{F}\ \text{get}) &\equiv \text{weak fairness} \\
\mathbf{A}(\mathbf{GF}\ \text{request} \rightarrow \mathbf{GF}\ \text{get}) &\equiv \text{strong fairness}
\end{aligned}$$

First-Order Logic is built from:

input symbols:	$R, p_1, \ldots, p_t$
variables:	$x, y, z, \ldots$
boolean connectives:	$\wedge, \vee, \neg$
quantifiers:	$\forall, \exists$
numeric symbols:	$=, \leq, 0, 1$

$$
\begin{aligned}
p(y) &\equiv p \\
(\exists y')(R(y,y') \wedge q(y')) &\equiv \mathbf{EX}q \\
(\forall y)p(y) &\sim \mathbf{AG}p \\
(\forall y)(\exists y')(p(y) \rightarrow (R(y,y') \wedge q(y))) &\sim \mathbf{AG}(p \rightarrow \mathbf{EX}q)
\end{aligned}
$$

Second-Order Logic is first-order logic plus relational variables which we may quantify.

$$
\begin{aligned}
\Phi \equiv\ & (\exists GYB)(\forall xy)((G(x) \vee Y(x) \vee B(x)) \wedge (R(x,y) \rightarrow \\
& (\neg(G(x) \wedge G(y)) \wedge \neg(Y(x) \wedge Y(y)) \wedge \neg(B(x) \wedge B(y))))
\end{aligned}
$$

FO(TC) = FO + Transitive Closure Operator

Given $\varphi(x_1, \ldots, x_k, x'_1, \ldots, x'_k)$, let

$$\mathbf{TC}_{\bar{x},\bar{x'}}(\varphi) \equiv \text{(the reflexive, transitive closure of } \varphi)$$

$$\mathbf{TC}^{s}_{\bar{x},\bar{x'}}(\varphi) \equiv \text{(the strict transitive closure of } \varphi)$$

$$\begin{aligned}\mathbf{TC}^{s}(\varphi)(y, y') \equiv\ & \mathbf{TC}(\varphi)(y, y')) \ \wedge\ (y \neq y' \ \vee \\ & (\exists y')(\varphi(y, y') \ \wedge\ \mathbf{TC}(\varphi)(y', y)))\end{aligned}$$

$$\begin{aligned}\mathbf{EF}p &\equiv \text{“We can get to a state where } p \text{ holds”} \\ &\equiv (\exists y')(\mathbf{TC}(R(y, y'))(y, y') \ \wedge\ p(y'))\end{aligned}$$

$$\begin{aligned}\mathbf{EG}p &\equiv \text{“Along some path p always holds”} \\ &\equiv (\exists y')(\mathbf{TC}(R(y, y') \wedge p(y))(y, y') \\ &\quad \wedge\ \ \mathbf{TC}^{s}(R(y, y') \wedge p(y))(y', y'))\end{aligned}$$

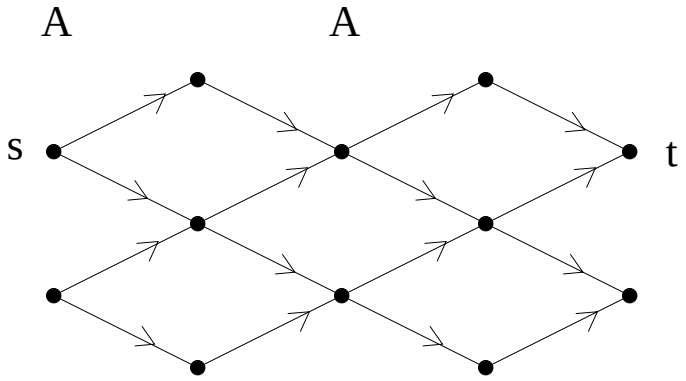
Fact: $\quad \text{NSPACE}[\log n] \quad = \quad \text{FO}(\mathbf{TC})$

Fact: P $=$ FO(**LFP**) $=$ FO + Inductive Definitions

$$\varphi(P,x,y) \equiv E(x,y) \vee (\exists z)(P(x,z) \wedge P(z,y))$$
$$\mathbf{TC}(E) = \mathbf{LFP}_{P,x,y}(\varphi) \quad = \quad \varphi^{\log n}(\emptyset)$$

In general, if φ is P^k-monotone

$$\mathbf{LFP}(\varphi) \quad \equiv \quad \min_P(\varphi(P) = P) \quad = \quad \varphi^{n^k}(\emptyset)$$



$$\varphi_{ap}(P,y) \equiv y = t \vee [(\exists y')(E(y,y') \wedge P(y')) \wedge (A(y) \rightarrow (\forall y')(E(y,y') \rightarrow P(y')))]$$
$$\mathrm{REACH}_a \equiv (\mathbf{LFP}_{Py}\varphi_{ap})(s)$$

The complexity of computing a query is closely tied to the complexity of describing the query.

Facts:

For constructible $t(n)$, $\mathrm{FO}[t(n)] = \mathrm{CRAM}[t(n)]$

For $k = 1, 2, \ldots,$ $\mathrm{VAR}[k+1] = \mathrm{DSPACE}[n^k]$

$$
\begin{aligned}
\mathrm{CRAM}[1] &= \mathrm{FO} \\
\mathrm{DSPACE}[\log n] &= \mathrm{FO}(\mathbf{DTC}) \\
\mathrm{NSPACE}[\log n] &= \mathrm{FO}(\mathbf{TC}) \\
\mathrm{P} &= \mathrm{FO}(\mathbf{LFP}) \\
\mathrm{NP} &= \mathrm{SO}\exists \\
\mathrm{PSPACE} &= \mathrm{SO}(\mathbf{TC}) = \mathrm{SO}(\mathbf{DTC}) = \mathrm{FO}(\mathbf{PFP}) \\
\mathrm{NSPACE}[n] &= \mathrm{SO}(\text{arity } 1)(\mathbf{TC}) \\
\mathrm{DSPACE}[n] &= \mathrm{SO}(\text{arity } 1)(\mathbf{DTC}) \\
\mathrm{ETIME} &= \mathrm{SO}(\text{arity } 1)(\mathbf{LFP})
\end{aligned}
$$

Data Complexity: fixed query, structure size $= n$

Modal μ-Calculus

$$
\begin{aligned}
\text{VAR} &\equiv \text{variables:} \quad \{Y, Z \ldots\} \\
\langle R \rangle &\equiv \text{next move modal operator} \quad \equiv \mathbf{EX} \\
\mu Z(\varphi(Z)) &\equiv \text{least fixed point:} \quad \mathbf{LFP}_{Z,y}(\varphi(Z, y)) \\
\nu Z(\varphi(Z)) &\equiv \neg \mu Z(\neg \varphi(\neg Z)) \quad \text{greatest fixed point}
\end{aligned}
$$

Example:

$$
\begin{aligned}
\mathbf{EF}p &\equiv \mu Z(p \vee \langle R \rangle Z) \\
&\equiv \mathbf{LFP}_{Z,y}(p(y) \vee \exists y'(R(y, y') \wedge Z(y')))\,(y)
\end{aligned}
$$

Fact: CTL* $\subseteq$ μ-Calculus $\subseteq$ FO^2(**LFP**^{1})

Model Checking: given K, φ, does $K \models \varphi$.

Model Checking CTL:	TIME$[\|K\| \cdot \|\varphi\|]$	[CE]
Model Checking CTL*:	PSPACE complete	[SC]

Model Checking the μ-Calculus:

Alternations: $\mu Y(\cdots \nu Z(\psi(Y, Z)) \cdots)$
$L\mu_k = \mu$-Calculus with $k-1$ alternations.
$L\mu_1 =$ alternation-free μ-Calculus

CTL $\subseteq$ $L\mu_1$: linear-time embedding
FCTL, ECTL $\subseteq$ $L\mu_2$: linear-time embedding
CTL* $\subseteq$ $L\mu_2$: exponential-size embedding

Model Checking $L\mu_1$:	TIME$[\|K\| \cdot \|\varphi\|]$	[CS]
Model Checking $L\mu_2$:	TIME$[(\|K\| \cdot \|\varphi\|)^2]$	[EL]
Model Checking $L\mu_k$:	TIME$[(\|K\| \cdot \|\varphi\|)^{2+k/2}]$	[LBCJM]

REACH$_a \in L\mu_1$
Model Checking $L\mu_1$ is PTIME complete.

Theorem [IV]: There is a linear-time map, from CTL formulas to equivalent formulas in $\mathrm{FO}^2(\mathbf{TC})$.

Examples:

$$\mathbf{EF}p \equiv p(y) \vee \exists y'(\mathbf{TC}(R)(y,y') \wedge p(y'))$$

$$\mathbf{EG}p \equiv \exists y'(\mathbf{TC}(\rho_p)(y,y') \wedge \mathbf{TC}^s(\rho_p)(y',y'))$$

$$\rho_p(y,y') \equiv p(y) \wedge R(y,y')$$

$$\mathbf{E}(p\mathbf{U}q) \equiv q(y) \vee \exists y'(\mathbf{TC}(\rho_p)(y,y') \wedge q(y'))$$

Theorem [IV]: There is a linear-time map, g, from CTL* formulas to equivalent formulas in FO$^2(\mathbf{TC})$.

Corollary[KVW]: CTL* model checking $\in$ NSPACE[$\log n$].

Problem: in extending the proof to CTL*:

$$\alpha \quad \equiv \quad \mathbf{E}(p \rightarrow q\mathbf{U}r)\mathbf{U}t$$

Solution: add a boolean b to remember status of $q\mathbf{U}r$.

$$\begin{aligned}
\alpha^\star \;&\equiv\; (\exists y'b')(\mathbf{TC}(\gamma)(y,0,y',b') \;\wedge\; t(y') \;\wedge \\
&\qquad\qquad b' \rightarrow (\exists y)(\mathbf{TC}(M_q)(y',y) \wedge r(y)) \\
\gamma(y,b,y',b') \;&\equiv\; ((p \vee b) \rightarrow (r \vee (q \wedge b'))) \;\wedge\; R(y,y') \\
M_q(y,y') \;&\equiv\; R(y,y') \;\wedge\; q(y)
\end{aligned}$$

A formula in FO^v or FO$^v(\mathbf{TC})$ has at most v domain variables, but may also have some **boolean** variables.

proof of $\mathrm{CTL}^\star \subseteq \mathrm{FO}^2(\mathbf{TC})$**:**

Add boolean b_γ for each $\gamma \in cl(\varphi)$.

$$
\begin{array}{lll}
\rho_\varphi(y,\bar{b},y',\overline{b'}) \equiv & R(y,y') & \\
& \wedge\ b_\alpha \rightarrow g(\alpha)(y) & \text{state } \alpha \\
& \wedge\ b_{\alpha\wedge\beta} \rightarrow b_\alpha \wedge b_\beta & \text{path } \alpha\wedge\beta \\
& \wedge\ b_{\alpha\vee\beta} \rightarrow b_\alpha \vee b_\beta & \text{path } \alpha\vee\beta \\
& \wedge\ b_{\mathbf{x}\alpha} \rightarrow b'_\alpha & \text{path } \mathbf{X}\alpha \\
& \wedge\ b_{\alpha\mathbf{u}\beta} \rightarrow b_\beta \vee (b_\alpha \wedge b'_{\alpha\mathbf{u}\beta}) & \text{path } \alpha\mathbf{U}\beta \\
& \wedge\ b_{\alpha\mathbf{b}\beta} \rightarrow b_\beta \wedge (b_\alpha \vee b'_{\alpha\mathbf{b}\beta}) & \text{path } \alpha\mathbf{B}\beta \\
& \wedge\ m'_{\alpha\mathbf{u}\beta} \rightarrow (m_{\alpha\mathbf{u}\beta} \vee b_\beta) & \text{path } \alpha\mathbf{U}\beta
\end{array}
$$

$$
\begin{array}{rl}
g(\mathbf{E}(\varphi)) = & (\exists y'\,\bar{b}\,\overline{b'}\,\overline{m})(b_\varphi \wedge \mathbf{TC}(\rho_\varphi)(y,\bar{b},\overline{0},y',\overline{b'},\overline{0}) \\
& \wedge\ \mathbf{TC}^s(\rho_\varphi)(y',\overline{b'},\overline{0},y',\overline{b'},\overline{m}) \\
& \wedge\ b'_{\alpha\mathbf{u}\beta} \rightarrow m'_{\alpha\mathbf{u}\beta} \quad \text{for any } \alpha\mathbf{U}\beta \in cl(\varphi)
\end{array}
$$

Model Checking = Query Evaluation

$$\mathcal{K} = (\{1, \ldots, n\}, R, p_1, \ldots, p_r)$$

Medium n: $\quad \mathrm{FO}^2(\mathbf{LFP}^1) \quad \supset \quad \mu$-Calculus

Large n: $\quad \mathrm{FO}^2(\mathbf{TC}^1) \quad \supset \quad \mathrm{CTL}^\star$

Huge n: Symbolic Model Checking ...

$$\mathcal{K}' = (2^{\{1,\ldots,n\}}, \underbrace{\varphi_R, \varphi_{p_1}, \ldots, \varphi_{p_r}}_{\text{coded as OBDD's}})$$

$$\begin{aligned} \mathrm{SO(unary)}(\mathbf{DTC}) &= \mathrm{DSPACE}[n] \\ \mathrm{SO(unary)}(\mathbf{TC}) &= \mathrm{NSPACE}[n] \\ \mathrm{SO(unary, alternation\text{-}free)}(\mathbf{LFP}) &= \mathrm{ETIME}[n] \end{aligned}$$

Model Checking is also related to maintenance of Integrity Constraints

Conclusions + Future Research

- Model Checking = Query Evaluation
- As a function of the size of the state space, choose a feasible query language.
- Investigate dynamic Model Checking.
- For which families of queries is symbolic model checking feasible?
- **Number of boolean variables:** may need linear number in worst case, but we believe only a small handful needed for "most, useful queries."
- We believe that $\mathrm{FO}^2(\mathbf{TC})$ **may be more efficient** than μ-calculus in many practical cases. $\mathrm{FO}^2(\mathbf{TC})$ **is more expressive than** $\mathrm{CTL}^\star$, e.g, can express the parity of the number of times an event has occured along a path.