SYMBOLIC GENERATION OF OPTIMAL DISCRETE CONTROL

Michel Sintzoff

Dept of Computing Science and Engineering

Université catholique de Louvain

`michel.sintzoff@uclouvain.be`

WG 2.1 Meeting, Namur

11-15 December 2006

I. PRELIMINARIES

- Problem

- Background

II. SYMBOLIC GENERATION

- Stratification

- Symbolic Iterative Terms

- Symbolic Semi-Algorithm

III. EVALUATION

- Illustration

- Complexity Analysis

- Discussion, Conclusion

I. PRELIMINARIES

I.1. Problem

Data

- *(Symbolic) action system :*

$$\begin{aligned} S &= \mathbf{do}\ A\ \mathbf{od} \\ &= \mathbf{do}\ A_0 \ [\!]\ \cdots\ [\!]\ A_{N-1}\ \mathbf{od}. \end{aligned}$$

- *Target predicate* Q.
- *Domain* X of states.
- *Index set* $I = \{0, \ldots, N-1\}$.

- For $i \in I$, the *action* A_i is a guarded command with an integer cost $w_i \geq 1$:

$$B_i(x) \to x := f_i(x) \ < w_i > .$$

A *policy* C for S is a tuple $(C_0, \cdots, C_{N-1})$ of predicates such that $C_i \Rightarrow B_i$ for each i.

A policy C' refines a policy C

$$\text{iff} \qquad \forall i \in I : C'_i \Rightarrow C_i.$$

$S{\downarrow}_C$ is S where each B_i is replaced by C_i.

A policy C is *optimal* for (S, Q) if Q is reached using paths by $S{\downarrow}_C$ with a minimum total cost.

Aim : Find the optimal policy for (S, Q).

II.2. Background

Basic Predicate Transformer :

It is a syntactical map $pre.S$ such that

$$
\begin{array}{rcl}
pre.(\mathbf{do}\ A\ \mathbf{od}).P & \equiv & \bigvee_{n \in \mathcal{N}} (pre.A)^n.P \\
(pre.A)^0.P & \equiv & P \\
(pre.A)^{n+1}.P & \equiv & pre.A.((pre.A)^n.P) \\
pre.([]_{i \in I}\ A_i).P & \equiv & \bigvee_{i \in I} pre.A_i.P \\
pre.(B_i \to x := e).P & \equiv & B_i \wedge P^x_e
\end{array}
$$

Graph : $G_S = (X, E_S, w)$ where

$$E_S = \{(x, i, f_i(x)) \mid B_i(x)\}, \quad w(i) = w_i$$

Paths by S : x_0 or $x_0 \cdots i_k x_k \cdots$
where $x_0 \in X, (x_{k-1}, i_k, x_k) \in E_S$ for $k = 1 \cdots$.

$Paths(S)$ is the set of paths by S.

$Paths(x, S)$: the paths by S beginning with x.
$Paths(S, x)$: the finite paths by S ending on x.
$Paths(S, Y) \doteq \bigcup_{x \in Y} Paths(S, x)$.
$Paths(x, S, Y) \doteq Paths(x, S) \cap Paths(S, Y)$.

The *concatenation* $p.p'$ of $p \in Paths(S, y)$ and $p' \in Paths(y, S)$ is p followed by the result of removing the initial y from p'.

Path Costs : for any finite path $p \in Paths(S, X)$,

$$\begin{array}{rcl} cost(x) & = & 0, \qquad cost(xiy) = w(i) \\ cost(p.(xiy)) & = & cost(p) + w(i) \end{array}$$

Optimality Domain D : for all $x \in X$,

$$D(x) \;\equiv\; (Paths(x,S,Q) \neq \emptyset) \;\equiv (pre.S.Q)(x)$$

Value Function $V : X \to \mathcal{N}$. If $D(x)$ then

$$V(x) \;=\; \min\{cost(p) \mid p \in Paths(x,S,Q)\}$$

$OptPaths(x,S,Q)$
$\doteq \{p \mid p \in Paths(x,S,Q) \wedge cost(p) = V(x)\}$

The *(weakest) optimal policy* for (S,Q) is the policy C for S such that, for all $x \in X$,

$$Paths(x, S{\downarrow_C}) \;=\; OptPaths(x,S,Q).$$

A Little Hierarchy of Predicate Transformers

$(pre.S.Q)(x) \equiv D(x) \equiv$
$(Paths(x, S, Q) \neq \emptyset) \equiv (OptPaths(x, S, Q) \neq \emptyset)$.

$(wp.S.Q)(x)$
$\equiv D(x) \wedge (Paths(x, S) = Paths(x, S, Q))$.

$(owp.S.Q)(x)$
$\equiv D(x) \wedge (Paths(x, S) = OptPaths(x, S, Q))$.
Read owp as "weakest optimality precondition".

So the weakest optimal policy C is the weakest policy for S such that

$$owp.S{\downarrow_C}.Q \equiv pre.S.Q.$$

Clearly $wp.S.P \Rightarrow pre.S.P$, $owp.S.P \Rightarrow wp.S.P$ and $D \equiv Q \vee \bigvee_i C_i$.

Classical, State-Based Method (e.g. Bellman):

(0) The domain X is assumed to be finite.

(1) Compute the value function V.

(2) The weakest optimal policy for (S, Q) is $C = (C_0, \ldots, C_{N-1})$ where, for all i and x,

$$C_i(x) \;\equiv\; B_i(x) \wedge (V(x) - V(f_i(x)) = w_i).$$

A *termination policy* is given by

$$C_i(x) \;\equiv\; B_i(x) \wedge (V(x) - V(f_i(x)) > 0)$$

where V is a variant function.

Complexity of the State-Based Generation

If X is finite, the complexity of this method is polynomial in $\#X$, thanks to efficient shortest-paths algorithms on finite graphs.

Challenges

(i) The number of states is exponential in the number of variables.

(ii) Infinite domains can't be handled.

II. SYMBOLIC GENERATION OF OPTIMAL POLICIES

II.1. Stratification into Level-Sets

(Optimality) Levels belong to the set

$$X_L \doteq \{n \mid n \in \mathcal{N} \wedge n < \#Rng(V)\}.$$

Optimality Radius : $\rho \doteq \sup X_L$.

Level-to-Value Bijection $V_L : X_L \to Rng(V)$:

$$\begin{aligned} V_L(0) &= 0 \\ V_L(n+1) &= \min_{x \in X}\{V(x) \mid V(x) > V_L(n)\}. \end{aligned}$$

State-to-Level Function $L : X \to X_L$:

$$\begin{aligned} V(x) &= V_L(L(x)) \\ \text{i.e.} \quad L(x) &= V_L^{-1}(V(x)). \end{aligned}$$

Sub-Domains : for $n \in X_L$,

$$D^{(n)}(x) \quad \equiv \quad D(x) \wedge L(x) \leq n,$$

Sub-Guards :

$$C_i^{(n)}(x) \quad \equiv \quad C_i(x) \wedge L(x) \leq n.$$

Domain Strata :

$$F^{(n)}(x) \quad \equiv \quad D(x) \wedge L(x) = n.$$

Guard Strata :

$$F_i^{(n)}(x) \quad \equiv \quad C_i(x) \wedge L(x) = n.$$

Strata are $f$ringes or $f$ronts of growing subsets.

Aim : to compute the optimal control guards

$$C_i \equiv \sup_{n \in X_L} \{C_i^{(n)}\}$$

iteratively.

II.2. Symbolic Iterative Terms These terms must not use computations on states.

Obvious Iterative Terms

$$D^{(0)} \equiv F^{(0)} \equiv Q, \qquad C_i^{(0)} \equiv F_i^{(0)} \equiv \mathbf{false},$$

$$D^{(n+1)} \equiv D^{(n)} \vee F^{(n+1)}, \qquad C_i^{(n+1)} \equiv C_i^{(n)} \vee F_i^{(n+1)},$$

$$F^{(n+1)} \equiv \bigvee_i F_i^{(n+1)}, \qquad C_i \equiv C_i^{(\rho)}$$

where $n, n+1 \in X_L$ and $i \in I$.

Iterative Terms for Guard Strata

Informally,

$$F_i^{(n+1)} \equiv \neg D^{(n)} \wedge pre.A_i.F^{(m)}$$

such that $V_L(n+1) - V_L(m) = w_i$, if feasible.

Let $opt_i : X_L \to Bool$, $g_i : X_L \to X_L$ such that

$$\begin{aligned} opt_i(n) &\equiv V_L(n) - w_i \in Rng(V_L), \\ opt_i(n) &\Rightarrow V_L(n) - V_L(\, g_i(n)) = w_i. \end{aligned}$$

So $opt_i(n)$ implies $g_i(n) = V_L^{-1}(V_L(n) - w_i) = m$.

Hence

$$\begin{aligned} F_i^{(n+1)} \equiv\ & \neg D^{(n)} \\ & \wedge opt_i(n+1) \wedge pre.A_i.F^{(g_i(n+1))}. \end{aligned}$$

Additional Iterative Terms :

$$
\begin{aligned}
& V_L(n+1) \\
= \quad & \min_{i,m:m\leq n} \left\{ \begin{array}{l} w_i + V_L(m) \\ \mid \neg D^{(n)} \wedge pre.A_i.F^{(m)} \not\equiv \mathbf{false} \end{array} \right\}
\end{aligned}
$$

$$
\begin{aligned}
& opt_i(n+1) \\
\equiv \quad & \bigvee_{m:m\leq n} V_L(n+1) - V_L(m) = w_i
\end{aligned}
$$

If $opt_i(n+1)$ holds then

$$
\begin{aligned}
& g_i(n+1) \\
= \quad & \min_{m:m\leq n} \{m \mid V_L(n+1) - V_L(m) = w_i\}
\end{aligned}
$$

Reduction of Symbolic Iterative Terms

A term is *subsidiary* if it is
- a predicate transformation,
- a satisfiability expression,
- a disjunction of equalities between costs,
- a minimization, or
- a supremum,

and does not occur in S or Q.

To *reduce* E is to replace each subsidiary sub-term E' in E by a symbolic expression of the result of evaluating E'.

Additional simplifications are welcome. In particular, if $id \equiv e$ has been derived where e is reduced then e may replace id in a reduced term.

In the semi-algorithm hereafter, all subsidiary sub-terms must be reduced. Thus $id \equiv E$ means id identifies $reduce(E)$.

II.3. Symbolic Semi-Algorithm $GenOpt$

begin

$V_L(0) = 0; \quad F^{(0)} \equiv Q; \quad D^{(0)} \equiv F^{(0)};$

for each i : $\quad F_i^{(0)} \equiv$ **false**; $C_i^{(0)} \equiv F_i^{(0)}$;

for n **from** 0 **while** $\neg D^{(n)} \wedge pre.A.D^{(n)} \not\equiv$ **false** :

for each i :

$$\left[\begin{array}{lll} F_i^{(n+1)} & \equiv & \neg D^{(n)} \\ & & \wedge opt_i(n+1) \wedge pre.A_i.F^{(g_i(n+1))}; \\ C_i^{(n+1)} & \equiv & C_i^{(n)} \vee F_i^{(n+1)}; \end{array}\right.$$

$$F^{(n+1)} \equiv \bigvee_i F_i^{(n+1)};$$

$$D^{(n+1)} \equiv D^{(n)} \vee F^{(n+1)};$$

for each i : $\quad C_i \equiv C_i^{(\sup Dom(V_L))}$

end

The iterative terms for $V_L(n+1), opt_i(n+1)$ and $g_i(n+1)$ have been given.

Correctness

By construction, if $GenOpt$ terminates then the resulting policy $(C_0, \cdots, C_{N-1})$ is the weakest optimal policy for (S, Q).

The semi-algorithm $GenOpt$ terminates iff

- the optimality radius ρ is finite, and
- the reductions terminate.

III. EVALUATION OF THE APPROACH

III.1. Illustration

$$
\begin{array}{rlll}
S = & \mathbf{do} & A_0 : \; x \geq 0 \rightarrow x := 4x & <26> \\
 & [\!] & A_1 : \; x \geq 0 \rightarrow x := x + 1 & <13> \\
 & \mathbf{od} & &
\end{array}
$$

$$X = \mathcal{R}, \quad Q \equiv 8 \leq x \leq 10.$$

The iterates for $n = 0, 1$ are

$$
\begin{array}{rlll}
V_L(0) = & 0 & & \\
F_0^{(0)} \equiv & F_1^{(0)} \equiv \mathbf{false}, & C_0^{(0)} \equiv C_1^{(0)} \equiv \mathbf{false} \\
F^{(0)} \equiv & 8 \leq x \leq 10, & D^{(0)} \equiv 8 \leq x \leq 10
\end{array}
$$

$$
\begin{array}{rlll}
V_L(1) = & 13 & \\
F_0^{(1)} \equiv & \mathbf{false}, & C_0^{(1)} \equiv \mathbf{false}, \\
F_1^{(1)} \equiv & 7 \leq x < 8, & C_1^{(1)} \equiv 7 \leq x < 8 \\
F^{(1)} \equiv & 7 \leq x < 8, & D^{(1)} \equiv 7 \leq x \leq 10
\end{array}
$$

For $n = 2$:

$$\begin{aligned}
V_L(2) = &\ 26 \\
F_0^{(2)} \equiv &\ 2 \leq x \leq 2.5, \quad C_0^{(2)} \equiv 2 \leq x \leq 2.5, \\
F_1^{(2)} \equiv &\ 6 \leq x < 7, \quad C_1^{(2)} \equiv 6 \leq x < 8, \\
F^{(2)} \equiv &\ 2 \leq x \leq 2.5 \\
&\ \vee 6 \leq x < 7 \\
D^{(2)} \equiv &\ 2 \leq x \leq 2.5 \\
&\ \vee 6 \leq x \leq 10
\end{aligned}$$

... and so on until $n = \rho = 6$.

Then $V_L(6) = 78$.

The optimal guards $C_0 \equiv C_0^{(6)}, C_1 \equiv C_1^{(6)}$ are

$$\begin{aligned} C_0 &\equiv 0.5 \leq x \leq 0.625 \vee 1.5 < x \leq 2.5 \\ C_1 &\equiv 0 \leq x \leq 0.5 \\ &\quad \vee 0.625 < x \leq 1.5 \vee 2.5 < x < 8. \end{aligned}$$

Notes

- $D \equiv 0 \leq x \leq 10 \equiv C_0 \vee C_1 \vee Q$.
- The intervals in C_0 and C_1 are interleaved.
- Non-determinism is possible; e.g. $x = 0.5$.
- To guess optimal policies is not easy.
- The domain is not denumerable.

III.2. Complexity of $GenOpt$

Notations

- $T(G)$ is the complexity of G.
- $T_{sat}(G)$ is the complexity of satisfiability expressions and predicate transformations in G.
- $f \in Poly(h)$ stands for $f \in \mathcal{O}(h^k)$.

Complexity of $GenOpt$

If ρ is finite then

$T(GenOpt) \ \in \ Poly(\rho + N + T_{sat}(GenOpt))$.

Efficiency of $GenOpt$ when ρ is finite

Since

$T(GenOpt) \ \in \ Poly(\rho + N + T_{sat}(GenOpt))$,

two good cases are defined as follows.

(i) X is finite
and $\rho + N + T_{sat}(GenOpt) \in Poly(\log \# X)$:
Then $T(GenOpt) \in Poly(\log \# X)$.

(ii) X is infinite
and $N + T_{sat}(GenOpt) \in Poly(\rho)$:
Then $T(GenOpt) \in Poly(\rho)$.

III.3. Complexity for Reachability Domains

Reachability Domains, Levels, Radius and Strata

Reachability domain :

$$D(x) \equiv (Paths(x, S, Q)) \equiv (pre.S.Q)(x).$$

Reachability level of x, for x such that $D(x)$:

$$L_R(x) \doteq \min\{nb_{edges}(p) \mid p \in Paths(x, S, Q)\}$$

where $nb_{edges}(p)$ is the number of occurrences of edge labels in a path p.

Reachability radius : $\rho_R \doteq \sup Rng(L_R)$.

Reachability sub-domains and strata :
for $n \in Rng(L_R)$,

$$D^{(n)}(x) \equiv D(x) \wedge L_R(x) \leq n,$$

$$F^{(n)}(x) \equiv D(x) \wedge L_R(x) = n.$$

Symbolic Semi-Algorithm $GenReach$ *:*

begin

$F^{(0)} \equiv Q; \;\; D^{(0)} \equiv F^{(0)};$

for n **from** 0 **while** $\neg D^{(n)} \wedge pre.A.F^{(n)} \not\equiv$ **false** :

$$\begin{aligned} F^{(n+1)} &\equiv \neg D^{(n)} \wedge pre.A.F^{(n)}; \\ D^{(n+1)} &\equiv D^{(n)} \vee F^{(n+1)}; \end{aligned}$$

$D \equiv D^{(\rho_R)}$

end

Complexity of $GenReach$: *If* ρ_R *is finite then*

$T(GenReach) \in Poly(\rho_R + N + T_{sat}(GenReach))$.

Two efficiency conditions for $GenReach$ are determined, as in the case of $GenOpt$.

III.4. Complexity of Optimality vs. Reachability

Let $M_w = \max_{i \in I}\{w_i\}$. For any $p \in Paths(S, Q)$,

$$nb_{edges}(p) \;\leq\; cost(p) \;\leq\; nb_{edges}(p) \times M_w.$$

Thus

$$L_R(x) \;\leq\; L(x) \;\leq\; L_R(x) \times M_w,$$

and, given $\rho_R = \sup Rng(L_R), \rho = \sup Rng(L)$,

$$\rho_R \;\leq\; \rho \;\leq\; \rho_R \times M_w.$$

Hence ρ_R is finite implies $\rho \in Poly(\rho_R + M_w)$.
Hence

$$\begin{aligned} & T(GenOpt) \\ \in \; & Poly(\rho_R + M_w + N + T_{sat}(GenOpt)). \end{aligned}$$

Case ρ_R and X are finite :

(A) If $\rho_R + M_w + N + T_{sat}(GenOpt) \in Poly(\log \# X)$
then $T(GenOpt) \in Poly(\log \# X)$
and $GenOpt$ is efficient.

(B) If $\rho_R + N + T_{sat}(GenReach) \in Poly(\log \# X)$
then $T(GenReach) \in Poly(\log \# X)$
and $GenReach$ is efficient.

Condition (A) for $GenOpt$ holds if

- Condition (B) for $GenReach$ holds,
- $M_w \in Poly(\rho_R)$,
- $T_{sat}(GenOpt) \in Poly(T_{sat}(GenReach))$.

Case ρ_R is finite and X is infinite : same conclusion.

III.5. Discussion

State-Based and Symbolic Greedy Algorithms

In Dijkstra's algorithm for shortest-path lengths, each iteration step
- computes the next optimal value, if needed,
- generates one new state with the considered optimal value.

In $GenOpt$, the iteration step for level $n+1$
- computes the optimal value $V_L(n+1)$,
- generates the set $F^{(n+1)}$ of all states having this optimal value.

Termination

In restricted families of symbolic transition systems, the termination of $GenReach$ is guaranteed (Henzinger, Majumdar, Raskin 05). Similar families could thus ensure the termination of $GenOpt$.

Efficiency

In case ρ_R is finite, $GenOpt$ is efficient provided
- $GenReach$ is efficient,
- $M_w \in Poly(\rho_R)$,
- $T_{sat}(GenOpt) \in Poly(T_{sat}(GenReach))$.

The efficiency of $GenReach$ is improved by
- simplification, e.g. abstraction,
- optimization, e.g. incrementalization,
- economical data structures.

Likewise for the efficiency of $GenOpt$.

From Value Functions to Optimal Policies

Usual approach: optimal policies are extracted from value functions, to be computed first.

Case of discrete-time finite-state systems : The value functions are produced using shortest-paths algorithms for finite graphs.

Case of timed automata :
The value functions are generated by algorithms which are symbolic wrt. clocks, thanks to the linearity of the considered continuous dynamics (Asarin, Maler 99; LaTorre et al. 02). But the extraction of optimal policies may be hard (Bouyer et al. 05).

From Optimal Policies to Value Functions

Optimal policies are generated by $GenOpt$ directly, without computing the value function.

Optimal costs may be obtained using the predicate $(opre.S.Q)(x,y) \equiv (D(x) \wedge y = V(x))$. This predicate is computed by adding the following iterative terms in $GenOpt$:

$$\begin{array}{rcl} W^{(0)} & \equiv & F^{(0)} \wedge y = 0 \\ W^{(n+1)} & \equiv & W^{(n)} \ \vee \ F^{(n+1)} \wedge y = V_L(n+1) \\ opre.S.Q & \equiv & W^{(\rho)}. \end{array}$$

If $V_L(n)$ reduces to v_n then $(opre.S.Q)(x,y)$ reduces to

$$\bigvee_{n \in X_L} F^{(n)}(x) \wedge y = v_n.$$

Thus, for a finite X_L, the guarded conditional

$$[]_{n \in X_L} F^{(n)}(x) \rightarrow y := v_n$$

implements $y = V(x)$ if $D(x)$.

This allows to compute shortest path-lengths for the following graphs, finite or not:

- the graphs are defined by action systems,
- the reachability radiuses are finite,
- the symbolic reductions terminate.

Value Functions vs Optimality Policies

As observed by Bellman, optimal policies and value functions f are dual to each other. He adds:

The purpose of our investigation is not so much to determine $f(x)$, which is really a by-product, but more importantly, to determine the structure of the optimal policy.

Verification of Optimality

To prove S is optimal from P to Q, one may prove $P \Rightarrow owp.S.Q$. Recall

$$\begin{aligned} & (owp.S.Q)(x) \\ \equiv\ & (pre.S.Q)(x) \\ & \wedge(Paths(x,S) = OptPaths(x,S,Q)). \end{aligned}$$

The predicate $owp.S.Q$ can be generated iteratively. For $n, n+1 \in X_L$,

$$\begin{aligned} K^{(0)} &\equiv Q, \quad H^{(0)} \equiv K^{(0)} \\ K^{(n+1)} &\equiv (\bigvee_{i\in I} B_i) \wedge \bigwedge_{i\in I}(B_i \Rightarrow opt_i(n+1) \wedge \\ &\qquad\qquad pre.A_i.K^{(g_i(n+1))}) \\ H^{(n+1)} &\equiv H^{(n)} \vee K^{(n+1)} \\ owp.S.Q &\equiv H^{(\rho)}. \end{aligned}$$

III.6. Further Work

Issues related to the proposed technique :

- improvement, implementation and application of $GenOpt$;
- problem classes for which the efficiency of $GenOpt$ is guaranteed;
- symbolic generation of reachability policies.

Other possible areas where optimal policies could be generated symbolically :

- games,
- probabilistic systems,
- continuous dynamics.

III.7. Conclusion

The procedures $GenReach$ and $GenOpt$ respectively generate reachability domains and optimal policies. $GenOpt$ refines $GenReach$ by taking action costs into account.

The synchronous iteration steps in $GenReach$ become asynchronous in $GenOpt$: optimal guard-strata for an action with a higher cost use domain strata with lower levels.

The efficiency of $GenOpt$ is guaranteed if
- $GenReach$ is efficient,
- the maximum action cost is polynomial in the reachability radius, and
- the complexity of reductions in $GenOpt$ is polynomial in that of reductions in $GenReach$.

$*\quad *$

$*$

A1. Synchronous vs. Asynchronous Iterations

The iterative term for $F_i^{(n+1)}$ defines an asynchronous iteration. Indeed

$$F_i^{(n+1)} \equiv \neg D^{(n)} \wedge pre.A_i.F^{(m)}$$

where m, if defined, may verify $m < n$.

If $\forall i : w_i = \mathbf{1}$ then $g_i(n+1) = m = n$ and the iteration is synchronous:

$$F_i^{(n+1)} \equiv \neg D^{(n)} \wedge pre.A_i.F^{(n)}$$

(vanLamsweerde, Sintzoff 1979).

A2. A Proof

$$
\begin{array}{ll}
 & F_i^{(n+1)}(x) \\
\equiv & \qquad \{ \text{ defn of } F_i^{(n)} \} \\
 & L(x) = n+1 \wedge C_i(x) \\
\equiv & \qquad \{ \text{ props of } C_i \} \\
 & L(x) = n+1 \wedge C_i(x) \wedge opt_i(n+1) \\
 & \wedge D(f_i(x)) \wedge L(f_i(x)) = g_i(n+1) \\
\equiv & \qquad \{ \text{ gen. of } C_i \text{ using } V\text{; defn of } g_i \} \\
 & L(x) = n+1 \wedge B_i(x) \wedge opt_i(n+1) \\
 & \wedge D(f_i(x)) \wedge L(f_i(x)) = g_i(n+1) \\
\equiv & \qquad \{ \text{ defn of } F^{(n)} \} \\
 & L(x) = n+1 \wedge B_i(x) \wedge opt_i(n+1) \\
 & \wedge F^{(\,g_i(n+1))}(f_i(x)) \\
\equiv & \qquad \{ \text{ defn of } pre \} \\
 & L(x) = n+1 \\
 & \wedge opt_i(n+1) \wedge pre.A_i.F^{(g_i(n+1))}(x) \\
\equiv & \qquad \{ \text{ defn of } D^{(n)} \} \\
 & \neg D^{(n)}(x) \\
 & \wedge opt_i(n+1) \wedge pre.A_i.F^{(g_i(n+1))}(x)
\end{array}
$$

A3. Control Policies as Relations

$$C_i(x) \equiv \pi(x, i) \equiv i \in \kappa(x)$$

where $\pi \subseteq X \times I$ and $\kappa : X \rightarrow 2^I$.

Non-deterministic (resp. deterministic) actions represent recurrence inclusions (resp. recurrence equations).