Regular Programs

Fischer / Ladner 1979 (Propositional dynamic logic)

Dexter Kozen 1997 (Kleene algebras with tests)

12-6 G.Smolka June 30, 2005

Goal: Prove Properties of Programs

- Programs are represented as ground terms in S
- Programs are interpreted as input/output relations
- Deduction of program equivalences from axioms

Example

$\boxed{\text{if } X \le 0 \text{ then } Z := 0 \text{ else } (Z := X * Y;\ X := 0)}$

$\boxed{Z := 0;\ \text{while } X \ge 0 \text{ do } (Z := Z + Y;\ X := X - 1)}$

In what sense are the 2 programs equivalent?

$X \in Loc$ locations

$\sigma \in \Sigma \overset{def}{=} Loc \to \mathbb{Z}$ states

Both programs describe the same function $\Sigma \to \Sigma$

initial state $\rightsquigarrow$ final state (input/output relation)

Simple Programs: Syntax

$z \in \mathbb{Z}$

$X \in Loc$

		interpretation
$a \in AE = z \mid X \mid a + a$	arithmetic expressions	$\Sigma \to \mathbb{Z}$
$b \in BE = a \le a \mid \neg b$	Boolean expressions	$\Sigma \to \mathbb{B}$
$p \in PS = X := a \mid p;p \mid \text{if } b \text{ then } p \text{ else } p \mid \text{while } b \text{ do } p$	simple programs	$\Sigma \rightharpoonup \Sigma$

Arithmetic Expressions in S

Constants		Standard interpretation $\mathcal{R}$
A		$\mathcal{R}A = \Sigma \to \mathbb{Z}$
$z : A$	for all $z \in \mathbb{Z}$	$\mathcal{R}z\,\sigma = z$
$x : A$	for all $x \in Loc$	$\mathcal{R}x\,\sigma = \sigma x$
$+ : A \to A \to A$		$\mathcal{R}+ f g\,\sigma = f\sigma + g\sigma$

Arithmetic expression $\widehat{=}$ ground term of type A

ground term : term not containing variables and λ's

Boolean Expressions in S

Constants	Standard interpretation $\mathcal{R}$
B	$\mathcal{R}B = \Sigma \to \mathbb{B}$
$\leq : A \to A \to B$	$\mathcal{R}\leq f g\,\sigma = (f\sigma \leq g\sigma)$
$\neg : B \to B$	$\mathcal{R}\neg f\sigma = \neg(f\sigma)$

Boolean expression $\widehat{=}$ ground term of type B

Regular Programs

- describe binary relations on Σ, not necessarily functional
- are mathematically easier than simple programs
- subsume simple programs (i.e., simple programs can be expressed as regular programs)
- have nondeterministic computational interpretation with 3 outcomes: success, failure, divergence

Regular Programs: Syntax

$p \in PR \;=\; x := a$	assignment
$\quad\mid\; b?$	test
$\quad\mid\; p;p$	sequential composition
$\quad\mid\; p+p$	choice
$\quad\mid\; p^*$	iteration

Regular Programs in $\mathcal{S}$

simple programs will follow soon

Constants	Standard interpretation $\mathcal{R}$
$\mathcal{P}$	$\mathcal{R}\mathcal{P} = \mathcal{P}(\Sigma^2)$
$? : \mathcal{B} \to \mathcal{P}$	$\mathcal{R}?f = \{(\sigma,\sigma) \in \Sigma^2 \mid f\sigma = 1\}$
$X{:=} : A \to \mathcal{P}$	$\mathcal{R}X{:=}\,g = \lambda\sigma\in\Sigma.\ \sigma[X := g\sigma]$
$; : \mathcal{P} \to \mathcal{P} \to \mathcal{P}$	$\mathcal{R}; R_1 R_2 = R_1 \circ R_2$
$+ : \mathcal{P} \to \mathcal{P} \to \mathcal{P}$	$\mathcal{R}+ R_1 R_2 = R_1 \cup R_2$
$* : \mathcal{P} \to \mathcal{P}$	$\mathcal{R}* R = Id\,\Sigma \cup R^+$

Regular program $\hat{=}$ ground term of type $\mathcal{P}$

Notational Conventions

- $;$ and $+$ are written infix where $;$ binds stronger than $+$
- $?$ and $*$ are written postfix

$$(b_1?;p_1 + b_2?;p_2)^* \rightsquigarrow$$

$$*(+\ \underbrace{(;(?b_1)\,p_1)}\ \underbrace{(;(?b_2)\,p_2)}\ )$$

More Notation

$\overline{b} \overset{def}{=} \neg b$

true $\overset{def}{=} 1 \leq 1$

false $\overset{def}{=} \neg$ true

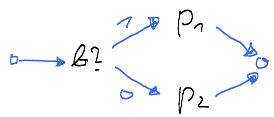
skip $\overset{def}{=}$ true?

fail $\overset{def}{=}$ false?

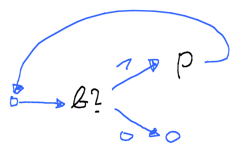
Simple Programs $\to$ Regular Programs

if b then p_1 else p_2 $\overset{def}{=} b?;p_1 + \overline{b}?;p_2$

while b do p $\overset{def}{=} (b?;p)^*;\overline{b}?$



if



while

Input / Output Relation

Let p be a regular program and ξ be some variable assignment.
Then $\mathcal{R}_p\xi \subseteq \Sigma \times \Sigma$ is called
the denotation of p or the input/output relation of p

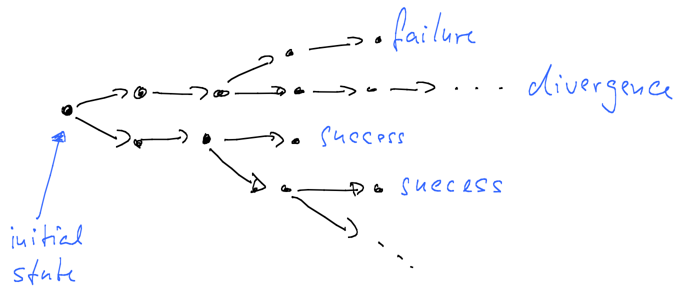
Since $\mathcal{R}_p\xi$ doesn't depend on ξ (there are no variables),
we will abuse notation and just write $\mathcal{R}_p$ for $\mathcal{R}_p\xi$

- A relation $R \subseteq \Sigma \times \Sigma$ is called
  - simple if $\exists\, p \in PS$: $R = \mathcal{R}_p$
  - regular if $\exists\, p \in PR$: $R = \mathcal{R}_p$

- A program $p \in PR$ is called functional if the relation $\mathcal{R}_p$ is functional

$\boxed{\text{Every simple program is functional}}$

Nondeterministic Computational Interpretation of Regular Programs



- maximal path $\hat{=}$ run of program
- success : regular termination
- failure : forced termination through failed test

Computational Intuitions

$b?$	fail if b not satisfied	
$x := a$	assign the value of a to x	
$p_1 ; p_2$	first p_1, then p_2	
$p_1 + p_2$	p_1 or p_2	nondeterministic
p^*	iterate p as long as you like	nondeterministic

Examples of Non-functional Programs

- $x := 1 + x := 2$ $\quad$ $\mathcal{R}_p$ non-functional, 2 outcomes
- $(x := x+1)^*$ $\quad$ $\mathcal{R}_p$ non-functional, ∞ outcomes

Execution Model for Regular Programs

1. If $\sigma \in Dom(\mathcal{R}p)$, then interpreter must yield one σ' such that $(\sigma,\sigma') \in \mathcal{R}p$; if there are several such σ', the choice is up to the interpreter.
2. If $\sigma \notin Dom(\mathcal{R}p)$, then interpreter must fail or diverge
3. Denotation $\mathcal{R}p$ doesn't distinguish between failure and divergence; hence we cannot be more specific in (2).

Computational Intuitions for Simple Programs

- denote functions $\Sigma \to \Sigma$
- $\exists$ diverging run of p on $\sigma \iff \sigma \notin Dom(\mathcal{R}p)$
- execution can avoid failure because of "simple" choices

$$\to + \begin{cases} \nearrow \bar{b}? \to \\ \searrow b? \to \end{cases} \quad \rightsquigarrow \quad \to b? \begin{cases} \xrightarrow{0} \\ \xrightarrow{1} \end{cases}$$

The Iter Notation

$$\text{iter } b_1 \Rightarrow p_1 \mid \cdots \mid b_n \Rightarrow p_n$$

$$\stackrel{def}{=}$$

$$(b_1?;p_1 + \cdots + b_n?;p_n)^*;\ \bar{b}_1?;\ \ldots;\ \bar{b}_n?$$

iterate as long as a rule is applicable

rule: $b \Rightarrow p$

guard action

Example: Coffee Beans Program

$$\boxed{\begin{array}{ll} \text{iter} & B \geq 1 \wedge W \geq 1 \Rightarrow B := B-1 \\ \mid & B \geq 2 \Rightarrow B := B-1 \\ \mid & W \geq 2 \Rightarrow W := W-2;\ B := B+1 \end{array}}$$

B: number of black beans
W: number of white beans

- not simple since guards are not mutually exclusive
- no diverging runs ($B+W \geq 0$ decreases)
- denotation is function $\Sigma \to \Sigma$ (will be shown later)
- conjunction in first guard can be eliminated

$$\begin{array}{l} (\ B \geq 1?;\ W \geq 1?;\ B := B-1 \\ +\ B \geq 2?;\ B := B-1 \\ +\ W \geq 2?;\ W := W-2;\ B := B+1)^*;\ B+W \leq 1? \end{array}$$

RP : Axioms for Regular Programs

RP is the set consisting of the following equations

Properties of ;

$p;(q;q') = (p;q);q'$	Ass
$p;skip = p = skip;p$	Ident
$p;fail = fail = fail;p$	Dom

Properties of +

$p+(q+q') = (p+q)+q'$	Ass
$p+q = q+p$	Com
$p+p = p$	Idem
$p+fail = p$	Ident

Distributivity

$p;(q+q') = p;q + p;q'$	Dist
$(q+q');p = q;p + q';p$	Dist

Properties of *

$p^*;p = p;p^*$	*;
$p^* = skip + p;p^*$	*skip;
$p^* = p^*;p^*$	*;*
$(b?)^* = skip$	?*

$\boxed{\mathcal{R} \models RP}$

all axioms are satisfied by standard interpretation

$\boxed{RP \vdash \text{while true do skip} = \text{fail}}$

Proof while true do skip

$= (true?;skip)^*;\ \overline{true}?$ while

$= (true?;skip)^*;\ false?$ false

$= (true?;skip)^*;\ fail$ fail

$= fail$ Dom □

$\boxed{RP \vdash \text{while } b \text{ do } p = \text{if } b \text{ then } p;\text{ while } b \text{ do } p \text{ else skip}}$

$b:\mathcal{B},\ p:\mathcal{P}$ are variables

Proof if b then p; while b do p else skip

$= b?;\ p;\ (b?;p)^*;\ \overline{b}? + \overline{b}?;skip$ if, while

$= b?;\ p;\ (b?;p)^*;\ \overline{b}? + skip;\ \overline{b}?$ Ident

$= (b?;\ p;\ (b?;p)^* + skip);\ \overline{b}?$ Ass, Dist

$= (b?;p)^*;\ \overline{b}?$ Com, *skip;

$=$ while b do p while □

Characterisation of Loops by Recursive Equations

$\boxed{q = \text{while } b \text{ do } p \quad \vdash^{RP}_{\circ} \quad q = \text{if } b \text{ then } p;q \text{ else skip}}$

$b:\mathcal{B},\ p:\mathcal{P},\ q:\mathcal{P}$ are variables

Proof. Immediate consequence of result on previous slide □

$\boxed{\text{Characteristic equation of } p^* \text{ is } q = skip + p;q}$

Proof. Follows from Axiom *skip; □

S provides for

often called semantics

- syntax of programs
- interpretation of programs (input/output relation)
- proof system for program equivalence
  - can be strengthened by adding axioms for arithmetic and Boolean expressions
  - needs some form of "induction principle" for proof of motivating equivalence on slide 3