Programme

- Prove non-decidable properties of TMs
  - Reduction technique
- Describe non-decidable properties of other universal formalisms
  - Chomsky grammars
  - Java
- Prove non-decidable properties of non-universal formalisms
  - Games
  - Context-free grammars

Reduction - applications

- **Theorem 11.7**
  Acc-Λ = {e(T) | Λ ∈ L(T) } is not recursive
  Proof: Show Acc ≤ Acc-Λ

- **Theorem**
  Let T_U denote the universal Turing Machine, then
  Uni-Acc = {e(w) | w ∈ L(T_U) } is not recursive

  Proof: Show Acc ≤ Uni-Acc

Reduction - applications

- **Theorem 11.8**
  AccSome = {e(T) | L(T) is nonempty} is not recursive
  Proof: Show Acc-Λ ≤ AccSome
  AccEver = {e(T) | L(T)' is empty} is not recursive
  Proof: Show Acc-Λ ≤ AccEver
  Subset = {e(T_1)e(T_2) | L(T_1) ⊆ L(T_2)} is not recursive
  Proof: Show Acc-Ever ≤ Subset

Rice’s Theorem - definition

- **Definition**
  A property of languages is said to be *nontrivial* iff it is satisfied by some but not all recursively enumerable languages

Nontrivial language properties

$\forall\, \Lambda \in L$

- $L = \emptyset$
- $L = \Sigma^*$

- L is finite
- L is regular
- All strings in L have even length

Rice's Theorem

- **Theorem 11.9**

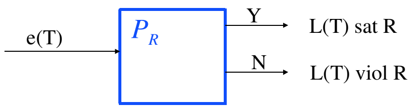
  Let R be any *nontrivial* property of languages, then

  P_R = {e(T) | L(T) has property R} is not recursive!

  Proof: Show Acc-$\Lambda \le P_R$

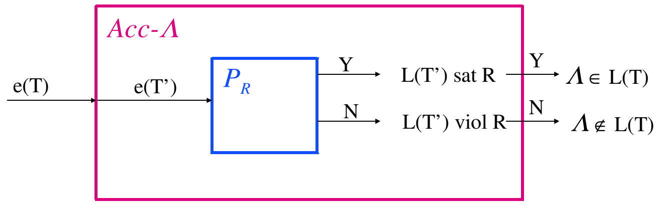
Reduction: *Acc-*$\Lambda \le P_R$

- Assume you had TM accepting P_R



Reduction: *Acc-*$\Lambda \le P_R$

- Construct TM accepting *Acc-*Λ



Assume $\emptyset$ viol R. Then T_R exists s.t. $L(T_R)$ sat R

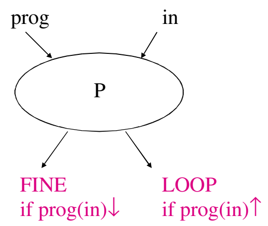
Construct T' s.t. if $\Lambda \notin L(T)$ then $L(T')=\emptyset$ else $L(T')= L(T_R)$

Programme

- Prove non-decidable properties of TMs
  - Reduction technique
- Describe non-decidable properties of other universal formalisms
  - Chomsky grammars
  - Java
- Prove non-decidable properties of non-universal formalisms
  - Context-free grammars

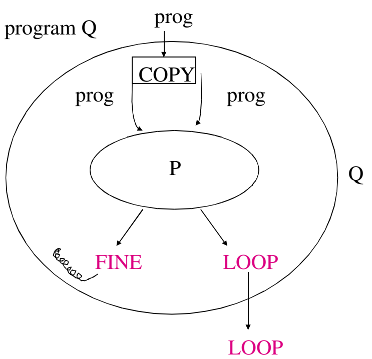
Harel diagonalization

Assume halting problem solvable in JAVA



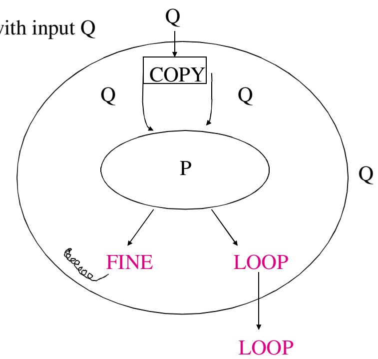
Harel diagonalization

Construct program Q



Harel diagonalization

Run Q with input Q



Busy Beaver

- **Definition**
  BB(n):= the maximal number of 1's printed by a Turing machine starting with n 1s on the tape and with n states
- **Exercise**
  BB is not computable!

Chomsky grammars

- A Chomsky grammar is a tuple
  G = (V, Σ, S, P), where

  V and Σ are finite disjoint sets of *variables* and *terminals* resp.
  S is the *start variable*, an element of V
  P is a set of *productions* of the form
  Type 3: $A \to a$ or $A \to aB$, where $A, B \in V$ and $a \in \Sigma$
  Type 2: $A \to \beta$, where $A \in V$ and $\beta \in (V \cup \Sigma)^*$
  Type 0: $\alpha \to \beta$, where $\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$ and $\beta \in (V \cup \Sigma)^*$

Chomsky type 0 languages

- Given a Chomsky type 0 grammar G = (V, Σ, S, P), define

  if $\alpha \to \beta \in P$, then for all $\alpha', \alpha'', \beta', \beta'' \in (V \cup \Sigma)^*$
  $$\alpha' \alpha \alpha'' \Rightarrow \beta' \beta \beta''$$

  $$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

Chomsky type 0 example

- Let G = (V, Σ, S, P), where
  V = {S, A, B, C}   Σ = {a, b, c}
  P: $S \to FT$

$T \to ABCT$	$T \to ABC$	
$BA \to AB$	$CA \to AC$	$CB \to BC$
$FA \to a$	$aA \to aa$	$aB \to ab$
$bB \to bb$	$bC \to bc$	$cC \to cc$

$$L(G) = \{a^i b^i c^i \mid i > 0\}$$

Turing and Chomsky

- **Theorems 10.8 and 10.9**
  For any language $L \subseteq \Sigma^*$,
  L is generated by a Chomsky type 0 grammar
  iff (constructively!!)
  L is accepted by a Turing machine

- **Corollary**
  All nontrivial properties of languages for Chomsky type 0 grammars are undecidable!

Programme

- Prove non-decidable properties of TMs
  - Reduction technique
- Describe non-decidable properties of other universal formalisms
  - Chomsky grammars
  - Java
- Prove non-decidable properties of non-universal formalisms
  - Games
  - Context-free grammars

Post's correspondence problem - example

- List A: List B:

List A		List B	
α_1 =	b	β_1 =	bbb
α_2 =	babbb	β_2 =	ba
α_3 =	ba	β_3 =	a

Does there exist a sequence of indices i1, i2,..., im $\in$ {1,2,3} such that

$$\alpha_{i1}\,\alpha_{i2}\ldots\ldots\,\alpha_{im} = \beta_{i1}\,\beta_{i2}\ldots\ldots\,\beta_{im}$$

Post's correspondence problem - example

List A		List B	
α_1 =	b	β_1 =	bbb
α_2 =	babbb	β_2 =	ba
α_3 =	ba	β_3 =	a

Solution?

Post's correspondence problem - example

- List A:
  α_1 = b
  α_2 = babbb
  α_3 = ba

  List B:
  β_1 = bbb
  β_2 = ba
  β_3 = a

Solution? YES: 2 1 1 3

$\alpha_2 \alpha_1 \alpha_1 \alpha_3$ = babbbbbba = $\beta_2 \beta_1 \beta_1 \beta_3$

Post's correspondence problem - example

- List A:
  α_1 = ba
  α_2 = abb
  α_3 = bab

  List B:
  β_1 = bab
  β_2 = bb
  β_3 = abb

Solution?

Post's correspondence problem - example

- List A:
  α_1 = ba
  α_2 = abb
  α_3 = bab

  List B:
  β_1 = bab
  β_2 = bb
  β_3 = abb

Solution? NO!

Post's correspondence problem - formally

- Given two finite lists of strings over some alphabet Γ
  List A: $\alpha_1, \alpha_2, .., \alpha_k$
  List B: $\beta_1, \beta_2, ..., \beta_k$
- Does there exist a sequence of indices i1, i2,...,im $\in$ {1,2,..,k} such that

  $\alpha_{i1} \alpha_{i2} ...... \alpha_{im} = \beta_{i1} \beta_{i2} ...... \beta_{im}$?

Modified Post's correspondence problem

- Given two finite lists of strings over some alphabet Γ
  List A: $\alpha_1, \alpha_2, .., \alpha_k$
  List B: $\beta_1, \beta_2, ..., \beta_k$

- Does there exist a sequence of indices
  i2, i3,...,im $\in$ {1,2,..,k} such that

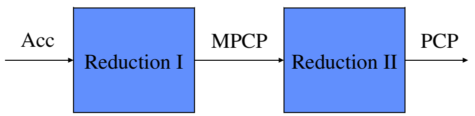
  $\alpha_1 \alpha_{i2} ...... \alpha_{im} = \beta_1 \beta_{i2} ...... \beta_{im}$?

Post's correspondence problem

- **Theorem**

  Post's correspondence problem is undecidable!

Post's correspondence problem - reductions



Modified Post's correspondence problem

- **Theorem 11.11**

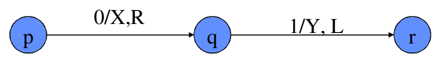
  Modified Post's correspondence problem is undecidable!

Reduction I

- Given Turing machine T and input w, construct *algoritmically* $MPCP_{T,w}$ such that

  T accepts w
  iff
  $MPCP_{T,w}$ has a solution

A Turing Machine



p010 |- Xq10 |- rXY0........

p010 # Xq10 # rXY0 #........

Reduction I

- Given $T = (Q, \Sigma, \Pi, \delta, q_0,)$ and $w \in \Sigma^*$
  (assume w.l.g. that T has no Stay-moves!)

- Alphabet of $MPCP_{T,w}$

  $\Gamma := (Q \cup \{h_a, h_r\}) \cup (\Pi \cup \{\Delta\}) \cup \{\#\}$

Reduction I - lists of $MPCP_{T,w}$

- List A: List B:
  $\alpha_1 = \#$ $\beta_1 = \#q_0\Delta w\#$

Reduction I - lists of $MPCP_{T,w}$

- List A: List B:

List A	List B	
$\alpha_1 = \#$	$\beta_1 = \#q_0\Delta w\#$	
$\alpha_d = X$	$\beta_d = X$	for all $X \in \Gamma$
$\alpha_{q,X} = qX$	$\beta_{q,X} = Yp$	if $\delta(q,X) = (p, Y, R)$
$\alpha_{q,X} = ZqX$	$\beta_{q,X} = pZY$	if $\delta(q,X) = (p, Y, L)$
$\alpha_{q,B} = q\#$	$\beta_{q,B} = Yp\#$	if $\delta(q, \Delta) = (p, Y, R)$
$\alpha_{q,B} = Zq\#$	$\beta_{q,B} = pZY\#$	if $\delta(q, \Delta) = (p, Y, L)$

Reduction I - lists of $MPCP_{T,w}$

- List A: List B:

List A	List B	
$\alpha_1 = \#$	$\beta_1 = \#q_0\Delta w\#$	
$\alpha_d = X$	$\beta_d = X$	for all $X \in \Gamma$
$\alpha_{q,X} = qX$	$\beta_{q,X} = Yp$	if $\delta(q,X) = (p, Y, R)$
$\alpha_{q,X} = ZqX$	$\beta_{q,X} = pZY$	if $\delta(q,X) = (p, Y, L)$
$\alpha_{q,B} = q\#$	$\beta_{q,B} = Yp\#$	if $\delta(q, \Delta) = (p, Y, R)$
$\alpha_{q,B} = Zq\#$	$\beta_{q,B} = pZY\#$	if $\delta(q, \Delta) = (p, Y, L)$
$\alpha_{a1} = Xh_aY$	$\beta_{a1} = h_a$	for all $X,Y \in \Gamma$
$\alpha_{a2} = Xh_a$	$\beta_{a2} = h_a$	for all $X,Y \in \Gamma$
$\alpha_{a3} = h_aY$	$\beta_{a3} = h_a$	for all $X,Y \in \Gamma$

Reduction I - lists of $MPCP_{T,w}$

- List A: List B:

List A	List B	
$\alpha_1 = \#$	$\beta_1 = \#q_0\Delta w\#$	
$\alpha_d = X$	$\beta_d = X$	for all $X \in \Gamma$
$\alpha_{q,X} = qX$	$\beta_{q,X} = Yp$	if $\delta(q,X) = (p, Y, R)$
$\alpha_{q,X} = ZqX$	$\beta_{q,X} = pZY$	if $\delta(q,X) = (p, Y, L)$
$\alpha_{q,B} = q\#$	$\beta_{q,B} = Yp\#$	if $\delta(q, \Delta) = (p, Y, R)$
$\alpha_{q,B} = Zq\#$	$\beta_{q,B} = pZY\#$	if $\delta(q, \Delta) = (p, Y, L)$
$\alpha_{a1} = Xh_aY$	$\beta_{a1} = h_a$	for all $X,Y \in \Gamma$
$\alpha_{a2} = Xh_a$	$\beta_{a2} = h_a$	for all $X,Y \in \Gamma$
$\alpha_{a3} = h_aY$	$\beta_{a3} = h_a$	for all $X,Y \in \Gamma$
$\alpha_s = h_a\#\#$	$\beta_s = \#$	

Reduction II

- Given MPCP over alphabet Γ
- Construct PCP over alphabet Γ' such that

  MPCP has solution
  iff
  PCP has solution

Definitions

$ir, il: \Gamma^* \rightarrow (\Gamma \cup \{\#\})^*$

$ir(\varepsilon) = \varepsilon \qquad ir(ax) = a\# \bullet ir(x) \qquad a \in \Gamma, x \in \Gamma^*$

$il(\varepsilon) = \varepsilon \qquad il(ax) = \#a \bullet il(x) \qquad a \in \Gamma, x \in \Gamma^*$

Examples:

ir(bob) = b#o#b#

il(bob) = #b#o#b

Reduction II

- Given MPCP with k lists over alphabet Γ

  A : $\alpha_1, \alpha_2, \ldots \alpha_k$ B : $\beta_1, \beta_2, \ldots \beta_k$

- Construct PCP with k+2 lists over $\Gamma \cup \{\#, \$\}$

  A': B':

  $\alpha'_0 = \#ir(\alpha_1)$ $\beta'_0 = il(\beta_1)$

  $\alpha'_i = ir(\alpha_1)$ $\beta'_i = il(\beta_i)$ for i = 1,2,..k

  $\alpha'_{k+1} = \$$ $\beta'_{k+1} = \#\$$

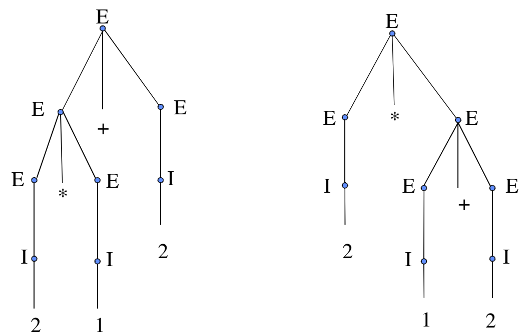
Context-free Grammar for expressions

- G = ({E, I}, {1,2,+,*,(,)}, P, S)

  P: E → I | E + E | E ∗ E | (E)

  I → 1 | 2

Ambiguity - example



Ambiguity - definition

- A grammar G is said to be *ambiguous* iff
  some string in L(G) has two different derivation trees

Unambiguous grammar for expressions

$E \to T \mid E + T$
$T \to F \mid T * F$
$F \to I \mid (E)$
$I \to 1 \mid 2 \mid I1 \mid I2$

$E \Rightarrow_G E+T \Rightarrow_G T+T \Rightarrow_G T * F+T \Rightarrow_G F*F+T$
$\Rightarrow_G I*F+T \Rightarrow_G 2*F+T \Rightarrow_G 2*I+T \Rightarrow_G 2*1+T$
$\Rightarrow_G 2*1+F \Rightarrow_G 2*1+I \Rightarrow_G 2*1+2$

Inherently ambiguous context-free language

- $L = \{a^n b^n c^m d^m \mid n,m \geq 1\} \cup \{a^n b^m c^m d^n \mid n,m \geq 1\}$
- (Ambiguous) grammar for L:

  $S \to AB \mid C$

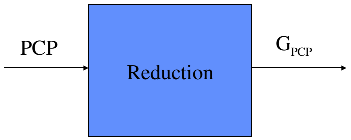
  $A \to aAb \mid ab$
  $B \to cBd \mid cd$

  $C \to aCd \mid aDb$
  $D \to bCd \mid bc$

Ambiguity problem for CFG's

- Given a context free grammar G = (V, Σ, P, S)
  Is G ambiguous?
- **Theorem 11.13**
  Ambiguity problem for CFG's is undecidable!

Ambiguity problem for CFG's - reduction



Construct context-free grammar G_{PCP} such that
PCP has solution iff G_{PCP} is ambiguous

Reduction

- Given PCP with lists A,B of k strings over alphabet Γ
- Construct G_{PCP} = $(\{S, A, B\}, \Delta \cup \{1,2,..k\}, P, S)$, where
  - $S \rightarrow A \mid B$
  - $A \rightarrow \alpha_1 A1 \mid \alpha_2 A2 \mid .. \mid \alpha_k Ak \mid$
    $\alpha_1 1 \mid \alpha_2 2 \mid .. \mid \alpha_k k$
  - $B \rightarrow \beta_1 B1 \mid \beta_2 B2 \mid .. \mid \beta_k Bk \mid$
    $\beta_1 1 \mid \beta_2 2 \mid .. \mid \beta_k k$

Undecidable problems for CFG's

- Given two context-free grammars G_1, G_2 over alphabet Σ
- Is $L(G_1) \cap L(G_2) = \emptyset$? *(Theorem 11.12)*
- Is $L(G_1) = \Sigma^*$ *(Theorem 11.15 - without proof!)*
- Is $L(G_1) = L(G_2)$? *(Exercise - use 11.15!)*
- Is $L(G_1)$ regular?

Programme

- Prove non-decidable properties of TMs
  - Reduction technique
- Describe non-decidable properties of other universal formalisms
  - Chomsky grammars
  - Java
- Prove non-decidable properties of non-universal formalisms
  - Games
  - Context-free grammars

Exercises

- Describe (un)decidability
  - 11.3 Reduction theorem for RE
  - 11.5 Fermat's last theorem
  - 11.15 A non-trivial problem solvable for TMs
  - 11.18 Example of PCP
- Explain algorithmic approaches to computability
  - 11.9 A TM reduction
  - 11.13 Unsolvable problems for C-programs
  - 11.19 PCP unsolvable for binary alphabets
  - 11.20 PCP solvable for unary alphabets
  - 11.21 Unsolvable problems for CFG (hint: use Thm 11.15)
  - Show that the Busy Beaver (slide 13) function is non-computable