CS 410/610, MTH 410/610
Theoretical Foundations of Computing

Fall Quarter 2010

Slides 5

Pascal Hitzler
Kno.e.sis Center
Wright State University, Dayton, OH
http://www.knoesis.org/pascal/

TOC: Undecidability

Chapter 12 of [Sudkamp 2006].

1. The Halting Problem
2. Problem Reduction (again)
3. Rice's Theorem
4. The Word Problem for Semi-Thue Systems
5. The Post Correspondence Problem
6. The Domino Problem

The Halting Problem

We know:

The language L_H = {R(M)w | M halts with input w} is recursively enumerable.

But is it recursive?

Differently put: Is the following problem decidable?

Given an arbitrary TM M with input alphabet Σ and a string $w \in \Sigma^*$, will the computation of M with input w halt?

This problem is called the *Halting Problem* (for Turing Machines).

Undecidability of the Halting Problem

Theorem 5.1

The Halting Problem (for TMs) is undecidable.

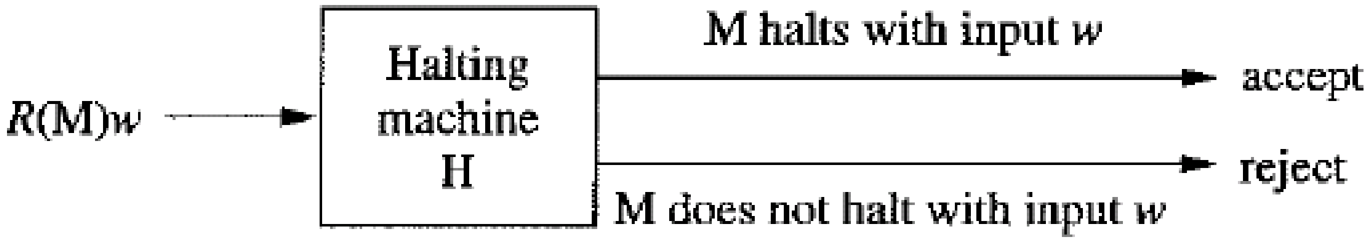
Proof idea:

Proof by contradiction, using diagonalization. But the diagonalization construction is a bit more complicated than usual.

Proof

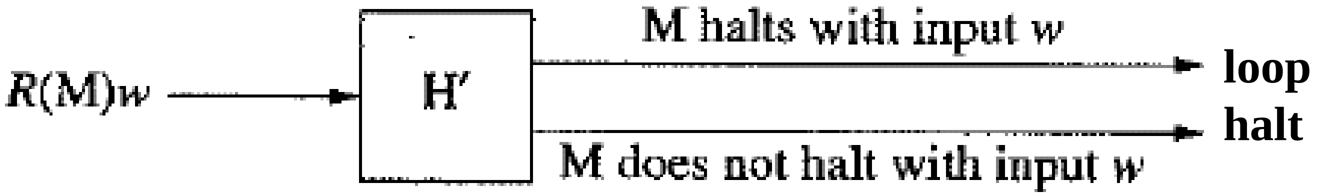
We use the universal TM approach from the previous chapter.

Assume there is a TM H which solves the Halting Problem.



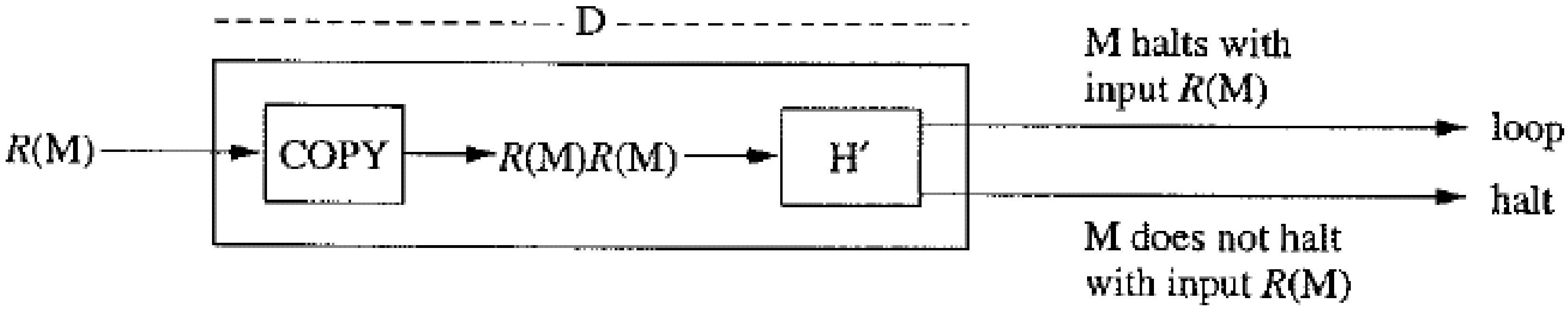
We now use H to construct an "impossible" machine D. This needs a few steps.

The TM H' does the same as H, except that it loops when H accepts. This is an easy modification of H.

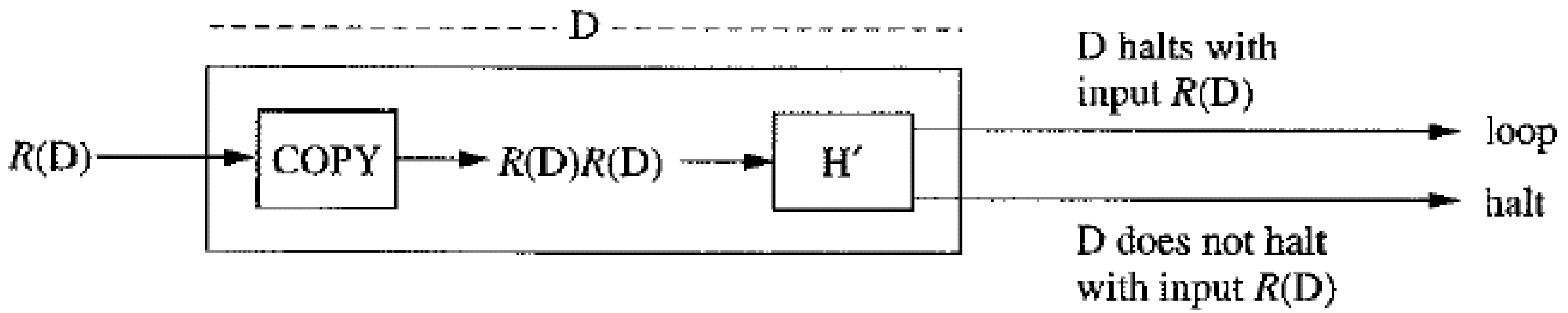


Proof

Use a TM COPY which copies a string (i.e. input u becomes output uu), together with H’ to construct a TM D:



Now consider a computation of D with input R(D):



Does D halt with input R(D)?

Proof

Thus, we have constructed a TM D, which halts on some input w=R(D) if and only if it does *not* halt on this input.

This is obviously nonsense. I.e. we have a contradiction in our assumptions.

Hence, we have to reject the initial assumption that there is a TM H which solves the Halting Problem.

This concludes the proof.

A different perspective on the proof

Diagonalization: Make a table, rows and columns entries are (all) TMs over {0,1} in some sequence M1, M2, M3, …

Entries in the table:

Row i, column j: 1 if Mi halts when run with R(Mj)
0 if MI does not halt when run with R(Mj)

The TM D is constructed such that it inverts the diagonal, i.e. D cannot be found in the table.

However, the table contains *all* TMs – and we have a contradiction.

Exercise 29 [no hand-in]

Show that the Halting Problem (for TMs) is semi-decidable.

The language L_H

Corollary 5.2

L_H = {R(M)w | M halts with input w} is not recursive.

Corollary 5.3

The recursive languages are a proper subset of the recursively enumerable languages.

Corollary 5.4

The language $\overline{L_H}$ is not recursively enumerable.

[Proof in Exercise 31.]

Corollary 5.5

There are undecidable problems which are not semi-decidable.

Exercise 30 [hand-in]

**Let Σ be an alphabet and L be a language over Σ.
Then we define $\overline{L} = \{w \in \Sigma^* \mid w \notin L\}$.**

Prove the following:

If L and $\overline{L}$ are recursively enumerable, then L is recursive.

Exercise 31 [hand-in]

Prove Corollary 5.4

Exercise 32 [no hand-in]

Formulate, in words, the decision problem captured in the language $\overline{L_H}$ (which is undecidable but not semi-decidable).

TOC: Undecidability

Chapter 12 of [Sudkamp 2006].

1. The Halting Problem
2. Problem Reduction (again)
3. Rice’s Theorem
4. The Word Problem for Semi-Thue Systems
5. The Post Correspondence Problem
6. The Domino Problem

Recall

Definition 4.1

Let L be a language over Σ_1 and Q be a language over Σ_2.

L is (*many-to-one*) *reducible* to Q
if there is a Turing computable function r: $\Sigma_1^* \to \Sigma_2^*$
such that w$\in$ L if, and only if, r(w)$\in$ Q.

Note: If L is reducible to Q, then

- **if Q is decidable, so is L.**
- **if Q is semi-decidable, so is L.**
- ***if L is undecidable, so is Q.***

Reduction revisited

We will often take a modified approach:

If we already know that a problem P is undecidable
and we want to show that a problem Q is undecidable

then:

we show that,
if Q were decidable
then P would also be decidable.

Essentially, this is a reduction approach – it's just that the explicit form mentioned in Definition 4.1 is not explicitly provided.

The Blank Tape Problem

Theorem 5.6

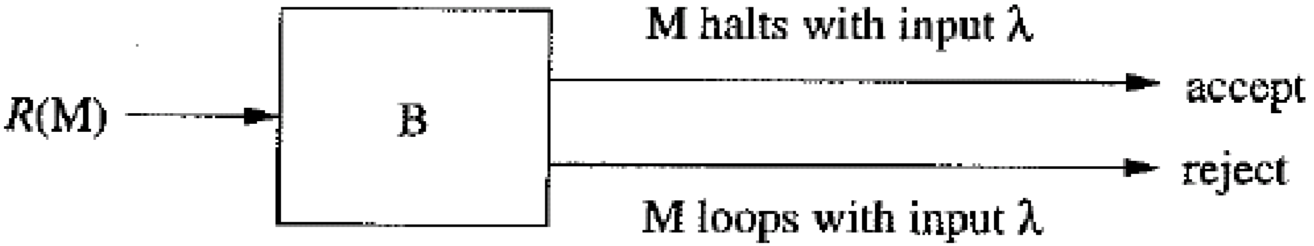
There is no algorithm that determines whether an arbitrary TM halts when a computation is initiated with a blank tape.

Proof idea?

Proof

We reduce the Halting Problem to the Blank Tape Problem.

Assume there is a TM B which solves the Blank Tape Problem:



The reduction (by a TM S) is as follows.

- S takes input of the form R(M)w
- S outputs a machine M' which does the following when run on an empty tape:
  - M' writes w on a blank tape and returns the tape head
  - M' simulates M (on the input w which is now on the tape).

Proof

Thus, if the Blank Tape Problem were decidable (say, by a TM B), then the Halting Problem would be decidable:

On input R(M)w, first run S, then run B.
If B accepts, then M halts on w.
If B rejects, then M does not halt on w.

Note:
S always terminates by definition.
B always terminates by assumption.

Note

The Blank Tape Problem is a *subproblem* of the Halting Problem, in the sense that it is obtained by fixing part of the input (namely the string w).

A subproblem of an undecidable problem may be decidable.
Example: Deciding, whether a particular TM halts on all inputs.
[We've done this several times.]

A subproblem of a decidable problem, however, is necessarily also decidable.
[Because in this case you can reduce the subproblem to the decidable problem.]

The Reenter Problem

Given M, w, does the TM M run on w reenter the start state?

We show this problem is undecidable by reduction of the Halting Problem.

Idea for the reduction?

Proof

Let M,w be input instances for the Halting Problem.

We construct M’ (which also runs on input w) which reenters its start state if, and only if, M halts when run with w.

We do this as follows:

- Add a new start state q_0’ for M’ (to make sure we only enter it when desired). It gets the same transitions as q_0.
- Add a transition to q_0’ for every halting configuration of M.

I.e.,

- The only way to reenter q_0’ is if M halts on w.
- Whenever M halts on w we will reenter q_0’.

The Halting On All Inputs Problem

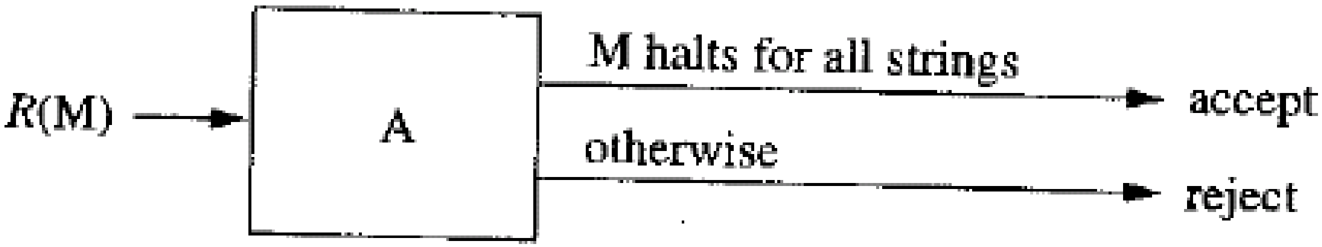
Determine, whether an arbitrary TM halts for *all* input strings.

This problem is also undecidable.
We show this by reduction of the Halting Problem.

Idea for the reduction?

Proof

Assume the TM A solves the Halting On All Inputs Problem:



The reduction is accomplished by a TM S:

1. **S determines whether the input string has the desired format R(M)w. If not, erase the tape (leave it blank). [leads to reject]**
2. **On input R(M)w, construct encoding of a TM M' that, when run on a string y,**
   - **erases y from the tape**
   - **writes w on the tape**
   - **runs M on w**

M' effectively ignores the input.

M' halts on any (on all) inputs if and only if M halts on input w.

TOC: Undecidability

Chapter 12 of [Sudkamp 2006].

1. **The Halting Problem**
2. **Problem Reduction (again)**
3. **Rice's Theorem**
4. **The Word Problem for Semi-Thue Systems**
5. **The Post Correspondence Problem**
6. **The Domino Problem**

Are undecidable problems “freaks”?

Say, decision problems which ask about properties of a recursively enumerable language L(M) accepted by a TM M (where M is the input to the problem), such as the following – are they likely to be decidable or undecidable?

- **Is $\lambda \in$ L(M)?**
- **Is L(M)=$\emptyset$?**
- **Does L(M) contain exactly 6 strings?**
- **Is L(M) = Σ*?**

These rephrased – are the following languages recursive?

- **L_λ = {R(M) | $\lambda \in$ L(M)}**
- **$L_\emptyset$ = {R(M) | L(M)=$\emptyset$}**
- **L_6 = {R(M) | L(M) has exactly 6 elements}**
- **$L_{\Sigma*}$ = {R(M) | L(M)=Σ*}**

Properties of languages

A *property* P of recursively enumerable languages describes a condition that a recursively enumerable language may satisfy, such as the following.

- The language contains λ.
- The language is the empty set.
- The language contains exactly 6 strings.

A property P of recursively enumerable languages is called *trivial* if there are no recursively enumerable languages that satisfy P or if every recursively enumerable language satisfies it.

A property which is not trivial is called *nontrivial*.

Properties of languages

A property P of recursively enumerable languages is called *trivial* if there are no recursively enumerable languages that satisfy P or if every recursively enumerable language satisfies it. A property which is not trivial is called *nontrivial*.

The following are nontrivial properties:

- The language contains λ.
- The language is the empty set.
- The language contains exactly 6 strings.
- The language contains all strings.

Exercise 33 [no hand-in]

Show that the following property is non-trivial:

L contains at least one string of even length.

Properties as languages

"L(M) has the property P"

can always be rephrased as

"Is $R(M) \in L_P$?"

for a suitably chosen L_P.

E.g.

- **The language L(M) contains λ.**
- **Is $R(M) \in L_\lambda$?**

(recall $L_\lambda = \{R(M) \mid \lambda \in L(M)\}$)

Rice's Theorem

Theorem 5.7

If P is a nontrivial property of recursively enumerable languages, then L_P is not recursive.

Proof idea?

Proof

Proof by reducing the Halting Problem to L_P.

Case 1: The empty language does not satisfy P.

Note there is at least one language $L \in L_P$, and $L \neq \emptyset$.
Let M_L be a TM that accepts L. [Why does M_L exist?]

Assume there is a TM M_P that decides membership in L_P.
We will now construct a solution to the Halting Problem.

Input: R(M)w
This is preprocessed into the encoding of a Machine M'.

Proof

M', when run with input y, does the following.

1. Write w to the right of y, producing ByBwB.
2. Run M on w.
3. If M halts when run with w, then run M_L with input y.

The result of running M' on y is exactly that of running M_L on y, provided M halts when run on w.
In this case, $L(M') = L(M_L) = L$ and $L(M')$ satisfies P.

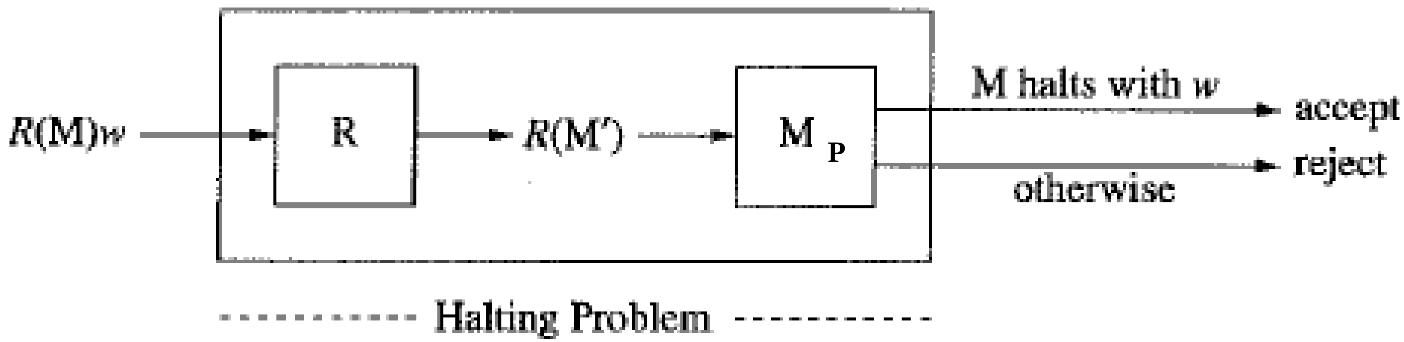
M' never halts on any input provided M does not halt on w.
In this case, $L(M') = \emptyset$, which does not satisfy P.

Thus, M' accepts $\emptyset$ when M does not halt with input w, and accepts L when M halts with w.

Thus, L(M') satisfies P if, and only if, M halts when run with input w.

Proof

Schemati



i.e. we have a solution to the Halting Problem, which is impossible.

Thus, P is not decidable.

Case 2: P is satisfied by the empty language.

Then use the preceding argument to show that $\overline{L_P}$ is not recursive.

By Exercise 34, L_P is not recursive.

Exercise 34 [hand-in]

Show that the following holds for all languages L:

If $\overline{L}$ is not recursive, then L is not recursive.

Exercise 35 [no hand-in]

Give an example of a property of languages that is not satisfied by any recursively enumerable language.

Exercise 36 [hand-in]

Is the following language recursive?
Is the following language recursively enumerable?

$L = \{a^i \mid i$ is a prime number$\}$

Prove your claims.

Exercise 37 [hand-in]

Give two different proofs that the following language is not recursive:

L = {R(M)w | M accepts w}

TOC: Undecidability

Chapter 12 of [Sudkamp 2006].

1. The Halting Problem
2. Problem Reduction (again)
3. Rice's Theorem
4. The Word Problem for Semi-Thue Systems
5. The Post Correspondence Problem
6. The Domino Problem

Semi-Thue Systems

- A type of grammar.
- Given an alphabet Σ.
- Define rewrite rules
  $u \to v$
  where $u \in \Sigma^+$, $v \in \Sigma^*$
- A rewrite rule replaces a substring u by a string v.
  - E.g., the rewrite rule
    $aaa \to bbbb$
    applied to the string ccaaac yields ccbbbbc
- The rewriting terminates if none of the rewrite rules is applicable.

- Given a Semi-Thue System $S = (\Sigma, P)$ – P is the set of rewrite rules of S – we write $u \Rightarrow_S v$ if, and only if, v can be obtained from u by exhaustive application of rules in S.

The Word Problem for S-T-Systems

The Word Problem for Semi-Thue-Systems is the problem of determining, for an arbitrary Semi-Thue System S and strings $u,v \in \Sigma^*$, whether v is derivable from u in S.

This is undecidable.

We will show this by reduction of the Halting Problem.

Proof idea?

Simulation

M = (Q,Σ,Γ,δ,q_0,F) a det. TM.

Define S-T-System S_M = (Σ_M, P_M) with
Σ_M = Q$\cup\Gamma\cup$ {[,],q_f, q_r, q_L} and the following rules:

1. $q_i xy \rightarrow zq_j y$ whenever $\delta(q_i, x) = [q_j, z, R]$ and $y \in \Gamma$
2. $q_i x] \rightarrow zq_j B]$ whenever $\delta(q_i, x) = [q_j, z, R]$
3. $yq_i x \rightarrow q_j yz$ whenever $\delta(q_i, x) = [q_j, z, L]$ and $y \in \Gamma$
4. $q_i x \rightarrow q_R$ if $\delta(q_i, x)$ is undefined
5. $q_R x \rightarrow q_R$ for $x \in \Gamma$
6. $q_R] \rightarrow q_L]$
7. $xq_L \rightarrow q_L$ for $x \in \Gamma$
8. $[q_L \rightarrow [q_f$.

Lemma 5.8

Let w = [uqv] be a string with u,v$\in\Gamma^*$ and q$\in$ Q$\cup$ {q_f, q_R, q_L}.

i) There is at most one string z such that w $\Rightarrow_{S_M}$ z.
ii) If there is such a z, then z also has the form [u'q'v'] with u',v'$\in\Gamma^*$, and q'$\in$ Q$\cup$ {q_f, q_R, q_L}.

Proof.

i) Determinism of M carries over to S_M.
ii) This follows easily from the form of the rules.

Lemma 5.9

A deterministic TM M halts with input w if, and only if, $[q_0BwB] \Rightarrow_{S_M} [q_f]$

Proof.

Note that a computation of M that halts with input w produces a derivation

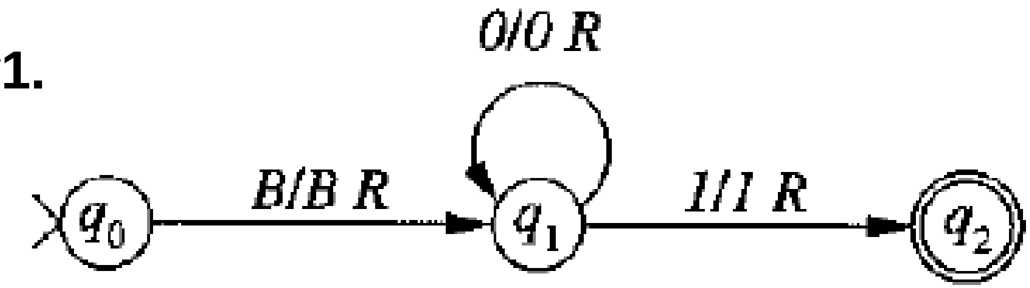
$$[q_0BwB] \Rightarrow_{S_M} [uq_Rv]$$

which is then transformed to $[q_f]$.

Conversely, a derivation of S_M to $[q_f]$ corresponds directly to a halting computation of M.

Example 5.10

The following TM accepts 0*1.



Rules of the corresponding Semi-Thue-System:

$q_0BB \to Bq_1B$	$q_10B \to 0q_1B$	$q_11B \to 1q_2B$
$q_0B0 \to Bq_10$	$q_100 \to 0q_10$	$q_110 \to 1q_20$
$q_0B1 \to Bq_11$	$q_101 \to 0q_11$	$q_111 \to 1q_21$
$q_0B] \to Bq_1B]$	$q_10] \to 0q_1B]$	$q_11] \to 1q_2B]$

$q_00 \to q_R$	$q_RB \to q_R$	$Bq_L \to q_L$
$q_01 \to q_R$	$q_R0 \to q_R$	$0q_L \to q_L$
$q_1B \to q_R$	$q_R1 \to q_R$	$1q_L \to q_L$
$q_2B \to q_R$	$q_R] \to q_L]$	$[q_L \to [q_f$
$q_20 \to q_R$		
$q_21 \to q_R$		

Exercise 38 [hand-in]

Trace

a) a computation of M from Example 5.10 on input 001.

b) a derivation of the corresponding S_M on the same input, in given in appropriate form.

Results

Theorem 5.11

The Word Problem for Semi-Thue Systems is undecidable.

Proof.

By reduction of the Halting Problem, as just given.

Theorem 5.12

Let M be a deterministic TM that accepts a nonrecursive language. Then the Word Problem for S_M is undecidable.

Proof.

Exercise 39 – more arguments than in the textbook, please, and you can use the referenced book-exercise only if you prove it as well.

Exercise 39 [hand-in]

See previous slide.

TOC: Undecidability

Chapter 12 of [Sudkamp 2006].

1. The Halting Problem
2. Problem Reduction (again)
3. Rice's Theorem
4. The Word Problem for Semi-Thue Systems
5. The Post Correspondence Problem
6. The Domino Problem

Dominoes

Given an alphabet Σ.
A domino is a pair d=(u,v) with $u,v \in \Sigma^*$.

A *Post correspondence system (Pcs)* consists of

- an alphabet Σ and
- a finite set of dominoes over Σ.

A solution to such a system is a sequence $d_1, \ldots, d_n$ of dominoes such that $u_1 \ldots u_n = v_1 \ldots v_n$.

The Post correspondence problem (Pcp) is the problem of determining whether a Post correspondence problem has a solution.

Example 5.11

With the domino types

a	c	ba	acb
ac	ba	a	b

a solution is given by

a	c	ba	a	acb
ac	ba	a	ac	b

Example 5.12

The Pcs with domino types [ab,aba], [bba,aa], [aba,bab] has no solution.

why?

We must start with [ab,aba].

As next domino, it must be either [ab,aba] or [aba,bab]. But with [ab,aba] there is a mismatch in the fourth element of the string. So we continue with [aba,bab].

We are then forced to continue with [bba,aa], which also produces a mismatch in the seventh element of the string.

We exhausted all possibilities, hence there is no solution.

Exercise 40 [no hand-in]

Show, that the PCS with domino types

[b,ba], [aa,b], [bab,aa], [ab,ba]

has no solution.

Undecidability Result

Theorem 5.13

The Pcp is undecidable.

Proof.

We show this by reduction of the word problem for Semi-Thue Systems.

Given a S-T-S S=(Σ,P) (Σ={0,1}) and a pair of strings u,v$\in\Sigma^*$, we construct a Pcs $C_{u,v}$ that has a solution if, and only if, u $\Rightarrow_S$ v.

Proof

Pcs alphabet: {0,0’,1,1’,(,),*,*’}

A string w consisting entirely of symbols with ’ is denoted w’.

Add to S the two redundant rules $0 \to 0$ and $1 \to 1$.

Dominoes:

[y_i’, x_i], [y_i, x_i’] for each $x_i \to y_i$ in S.

[(u*, ( ], [*, *’], [*’, *], [ ), *’v)]

[0,0’], [0’,0], [1,1’], [1’,1] (*)

we use an abbreviation: [w, w’] for any $w \in \{0,1\}^*$ stands for the corresponding sequence of dominos from row (*).

Proof

part 1: Assume u $\Rightarrow_S$ v. Then $C_{u,v}$ has a solution.

Note there is a derivation $u = u_0 \Rightarrow u_1 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$ **of even length.**

The i-th step of the derivation can be written

$$u_{i-1} = p_{i-1}x_{j_{i-1}}q_{i-1} \Rightarrow p_{i-1}y_{j_{i-1}}q_{i-1} = u_i$$

where u_i is obtained from u_{i-1} by application of $x_{j_{i-1}} \to y_{j_{i-1}}$ **.**

We now construct a solution to $C_{u,v}$ which is going to be the string

$$(u_0 * u_1' *' u_2 * u_3' *' \ldots * u_{k-1} *' u_k).$$

Proof

(Solution: (u_0 * u_1' *' u_2 * u_3' *' … * u_{k-1} *' u_k))

Start solution with [(u*, ( ].

Then match u at the bottom, in the form of

[(u*, ( ] , [p_0',p_0], [y_{j0}', x_{j0}], [q_0', q_0], [*', *]

where the dominoes spelling p_0 and q_0 are composite ones. The domino [y_{j0}', x_{j0}] comes from the rule $x_{j_0} \rightarrow y_{j_0}$.

This process is repeated with the first elements, producing (u_0 * u_1' *' u_2 * on the top, and then continued for the full derivation, yielding

(u_0 * u_1' *' u_2 * u_3' *' … * u_{k-1} *' u_k).

The sequence is completed with the domino [ ), *'v)].

Proof

part 2: Assume $C_{u,v}$ has a solution. Then u $\Rightarrow_S$ v.

Note the solution must start with [(u*,[ ) and must end with [ ),*'v)].

Thus the solution string has the form (u * w *' v). If w contains), then the shorter string ending with) is also a solution – so assume that) occurs only as the rightmost symbol in the solution.

The way the dominos are made, the solution defines a sequence of strings (u_0 * u_1' *' u_2 * u_3' *' … * u_{k-1} *' u_k), and each u_{i+1} can be obtained from u_i by a finite number of applications of rewrite rules. Thus, u $\Rightarrow_S$ v.

Part 1 and part 2 combined constitute a reduction of the word problem for S-T-Ss to the Pcp. The Pcp is thus undecidable.

TOC: Undecidability

Chapter 12 of [Sudkamp 2006].

1. The Halting Problem
2. Problem Reduction (again)
3. Rice’s Theorem
4. The Word Problem for Semi-Thue Systems
5. The Post Correspondence Problem
6. The Domino Problem

The Domino Problem

- Most prominently used undecidable problem for showing that certain Description Logics are undecidable.
  - Thus related to Semantic Web research.
    Context: See Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, Foundations of Semantic Web Technologies, CRC Press, 2010, http://www.semantic-web-book.org

- This is not in [Sudkamp 2006].

  Source: Lecture by Franz Baader, Complexity and Logic, Winter Semester 2005/2006, TU Dresden, Germany
  – through a manuscript on the lecture by Adila Alfa Krisnadhi

  See e.g. Börger, Grädel, Gurevich, The Classical Decision Problem, Springer, 2001.

Dominoes again

Note: This is a different domino setting than for the Pcp!

Dominoes are squares with colored sides with an orientation (they cannot be turned or rotated).

Question: given a finite set of domino types (unlimited supply each), can we tile a quarter plane such that touching sides always have the same color?

A domino system D = (S,H,V) consists of

- a finite set of domino types S
- a horizontal compatibility relation $H \sqsubseteq S \times S$
- a vertical compatibility relation $V \sqsubseteq S \times S$

Dominoes again

A domino system D = (S,H,V) consists of

- a finite set of domino types S
- a horizontal compatibility relation H $\sqsubseteq$ S $\times$ S
- a vertical compatibility relation V $\sqsubseteq$ S $\times$ S

A solution (tiling) of a domino system D=(S,H,V) is a mapping T: N $\times$ N$\to$ D (where N are the non-negative integers) such that

- if T(x,y)=d and T(x+1,y)=d', then (d,d')$\in$ H
- if T(x,y) = d and T(x,y+1) = d', then (d,d')$\in$ V

The Domino Problem

Theorem 5.14 (The Domino Problem)

The following problem is, in general, undecidable.

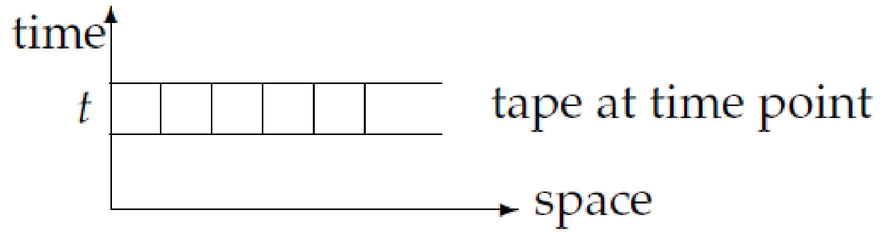
Given: a domino system D=(S,H,V) and $d_0 \in S$
Question: Does D have a solution T with $T(0,0)=d_0$?

Proof: By reduction of the Halting Problem.

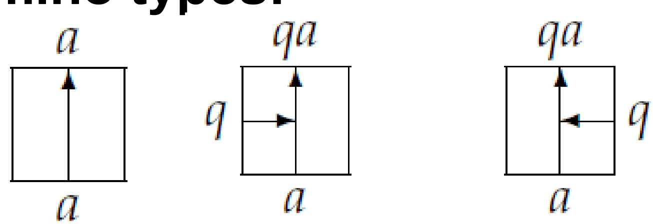
Given a TM M and a word w
we construct a domino system D=(S,H,V) and $d_0 \in S$ s.t. M halts on w if, and only if, D does *not* have a solution T with $T(0,0)=d_0$.

Proof

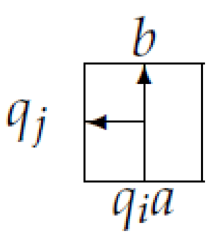
Idea: tiling represents space-time diagram of a computation of M.



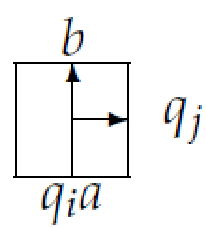
Domino types:



for all q∈ Q and a∈Γ

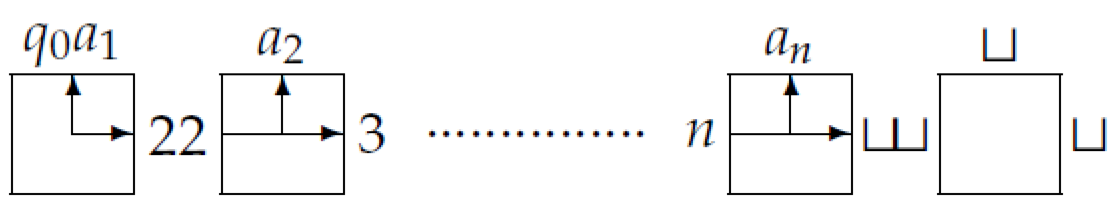


if $\delta(q_i, a) = (q_j, b, L)$



if $\delta(q_i, a) = (q_j, b, R)$

(q_0 =)



input encoding

Proof

The only way for the dominoes to fit together is by each row encoding a current configuration (tape + tape head + state) of the TM during a computation.

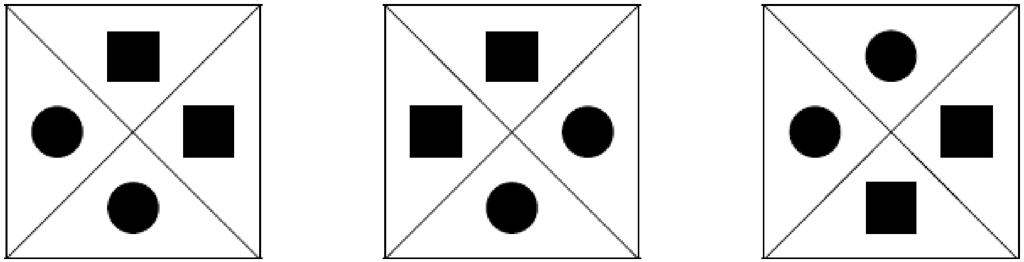
Moving up one row corresponds exactly to the application of one (applicable) transition.

The quarter plane can be filled entirely if, and only if, the computation does not terminate.

Thus, the computation terminates if, and only if, the Domino system does *not* have a solution.

Exercise 41 [no hand-in]

Does the following Domino system have a solution? (Pick any starting domino.)



What's next

Chapter 13 of [Sudkamp 2006]

μ-recursive functions

as another paradigm for computation.