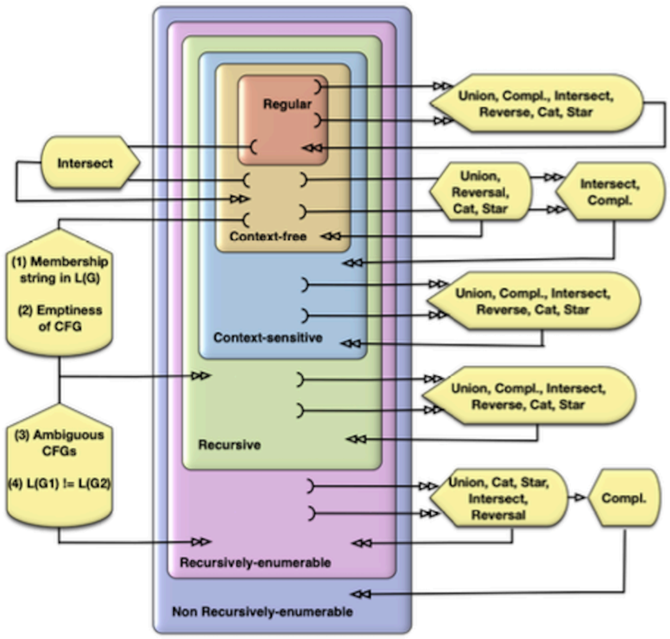
Closure of Languages



Reduction

A can be reduced to B:

1. A is no harder than B
2. B is easy $\Rightarrow$ A is easy
3. A is hard $\Rightarrow$ B is hard

Computability

A language L is recognizable (recursively enumerable) iff $\exists$ recognizer T s.t. $L(T) = L$.

- Recognizer either halts and accepts, halts and rejects, or spins indefinitely and rejects.

A language L is decidable (recursive) iff $\exists$ decider T s.t. $L(T) = L$.

- Decider either halts and accepts, or halts and rejects.

If a language L is recognizable, and $\overline{L}$ is recognizable, then L is decidable:

1. Construct TM M deciding L:
   - On input w:
     1. Simulate one step of the turing machine recognizing L
     2. Simulate one step of the turing machine recognizing $\overline{L}$
     3. If the first machine accepts, accept
     4. If the first machine accepts, reject
     5. Otherwise, repeat at (1)

M will accept if $w \in L$, and reject if $w \notin L$. It will never keep spinning, because one of these machines *will* accept.

Important Problems in Computability

<u>Shit we covered in class</u>:

1. A_{DFA}: Acceptance problem for DFAs; *decidable*
2. A_{NFA}: Acceptance problem for NFAs; *decidable*
   - A_{NFA} = "On input $\langle B, w\rangle$:
     1. Convert NFA B to equivalent DFA B'
     2. Run TM $D_{\text{A-DFA}}$ on input $\langle B', w\rangle$.
     3. If $D_{\text{A-DFA}}$ accepts, accept; otherwise, reject.
3. A_{REX}: Acceptance problem for regular expressions; *decidable*
   - convert regex to NFA, run $D_{\text{A-NFA}}$ on it
4. E_{DFA}: Emptiness problem for DFAs; *decidable*
5. EQ_{DFA}: Equivalence problem for DFAs; *decidable*
   - construct DFA for the symmetric difference of the two langauges, which we know we can do by properties of regular languages
6. A_{CFG}: Acceptance problem for CFGs; *decidable*
7. E_{CFG}: Emptiness problem for CFGs; *decidable*
8. EQ_{CFG}: Equivalence problem for CFGs; *not decidable*
   - can't use the same strategy as EQ for DFAs, because CFLs are not closed under complementation or intersection
9. A_{TM}: Acceptance problem for TMs; *not decidable*
   - $A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$
   - Assume some TM H decides A_{TM}.
   - Construct TM D: D = "On input $\langle M\rangle$:
     1. Run H on input $\langle M, \langle M\rangle\rangle$.
     2. Accept if H rejects, reject if H accepts.
   - D accepts $\langle M\rangle$ iff M doesn't accept any $\langle M\rangle$, for any M.
   - Let $M = D$: D accepts $\langle D\rangle$ iff D doesn't accept $\langle D\rangle$. This is a contradiction, and so our initial assumption is false; A_{TM} is undecidable.
10. $\overline{A_{\text{TM}}}$: Not Turing-recognizable.
    - Assume that $\overline{A_{\text{TM}}}$ is Turing-recognizable.
    - We know that A_{TM} is Turing-recognizable.
    - But, because if L and $\overline{L}$ is recognizable, then L is decidable, this implies A_{TM} is decidable.
    - But this contradicts what we know, that A_{TM} is not decidable. Therefore, $\overline{A_{\text{TM}}}$ is not Turing-recognizable.
11. $HALT_{\text{TM}}$: Undecidable.
    - $HALT_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$
    - Assume that $HALT_{\text{TM}}$ is decidable; then, $\exists$ TM R s.t. R decides $HALT_{\text{TM}}$.
    - Construct a new machine, S = "On input $\langle M, w\rangle$:
      1. Run R on $\langle M, w\rangle$.
      2. If R rejects - that is, if M spins forever on w - reject.
      3. Otherwise, M will halt. Simulate M on w; if M accepts, accept, and if M rejects, then reject.
    - This machine decides A_{TM}: For any Turing Machine M and string w, it decides the output of M on w. This is a contradiction, because we know that A_{TM} isn't decidable, and so our initial assumption that $HALT_{\text{TM}}$ is decidable is false.
12. E_{TM} is undecidable.
    - $E_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$
    - Assume that E_{TM} is decidable. Then, $\exists$ TM R s.t. R decides E_{TM}.
    - We can then construct S that decides A_{TM}.
    - $S =$ On input $\langle M, w\rangle$:
      1. Construct a new TM, M_w = On input x:
         1. If $x \neq w$, reject.
         2. If $x = w$:
            - Simulate M on w. Accept if M accepts.
         3. Run R on M_w.
         4. If R accepts, then $L(M_w) = \emptyset$, and M rejects w. Reject.
         5. Otherwise, then $L(M_w) = \{w\}$, and M accepts w. Accept.
    - S decides A_{TM}, which is a contradiction, yada yada, you get the picture.
13. $REGULAR_{\text{TM}}$ is undecidable.
    - $REGULAR_{\text{TM}} = \{\langle M\rangle \mid L(M) \text{ is regular}\}$
14. EQ_{TM} is undecidable.
    - $EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$.
    - Assume that EQ_{TM} is decidable. Then, $\exists$ TM R deciding EQ_{TM}.
    - We can construct S deciding E_{TM}:
    - $S =$ On input $\langle M\rangle$:
      1. Construct a TM M' s.t. $L(M') = \emptyset$
      2. Run R on $\langle M, M'\rangle$
      3. Accept if R accepts; otherwise, reject.
    - S decides E_{TM}, contradiction,f a;osihopa 94hoaeirth

Complexity

- P: The set of problems with a polynomial-time solution
- NP: The set of problems with a nondeterministic polynomial-time solution; equivalent to the set of problems whose solutions are verifiable in polynomial time
- NPC: The set of problems in NP-Hard and NP
- NP-Hard: A problem B is in NP-Hard iff one of the following conditions is true:
  1. $\forall$ problem $A \in$ NP, $A \leq_p B$; that is, A can be reduced to B in polynomial time.
  2. $\exists$ NP-Hard problem $A\ s.t.\ A \leq_p B$.