# Predicate Calculus
# - Undecidability of predicate calculus

Moonzoo Kim
CS Dept. KAIST

moonzoo@cs.kaist.ac.kr

# Undecidable problems

- Can you tell whether or not the following program halts?

```
/* Fermat's last theorem: for n > 2, there
   exists no positive integers x,y,z s.t. x^n +
   y^n = z^n */
main() {
    Nat n, total, x, y, z;
    scanf('%d",&n);
    total=3;
    while(1) {/* loop invariant:total= x+y+z*/
        for(x=1; x<= total-2; x++) {
            for(y=1; y <= total-x-1; y++) {
                z= total - x -y;
                if(x^n + y^n == z^n) halt;
            }
        }
        total++;}}
```

- It would be remarkable indeed if we could make an algorithm that could examine any program P and tell whether P would halt.
  - In other words, to decide whether a given program halts or not is, at least, as hard as proving the Fermat's last theorem which took 300 years
- We know that no such algorithm exists –
  - Halting problem is undecidable

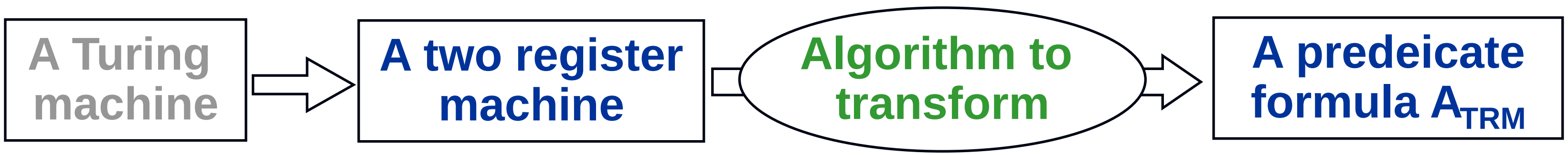
# Transform of the Halting problem (1/2)

- It is undecidable to check whether a Turing machine (TM) will halt if started on a blank tape (halting problem)
- To prove the undecidability of predicate logic, we give an algorithm which produces a formula $A_{TM}$ in the predicate calculus for every Turing machine, s.t. $A_{TM}$ is valid iff a Turing machine halts
  - Note that we do not make a Turing machine M for every predicate formula, since it is enough to show that checking some predicate formulas is undecidable
- If we have such an algorithm, it is clear that validity check of predicate formula is at least as hard as halting problem (i.e., undecideble)


**TM halts $\leftrightarrow$ $A_{TM}$ is valid**

# Transform of the Halting problem (2/2)

- To simplify the proof of the transformation algorithm, we work with two-register machines (TRM) rather than directly with Turing machine
  - i.e., we will show there exists such $A_{TRM}$ for a two-register machine
- Thm 5.42 Given a Turing machine that computes a function f, a two-register machine can be constructed to compute the same function f




*Thm 5.42*
*TM halts $\leftrightarrow$ TRM halts*

**TRM halts $\leftrightarrow$ $A_{TRM}$ is valid**

# A two-register machine M

- Def 5.41 A two-register machine M consists of two registers x and y which can hold natural numbers, and a program $P = (L_0,\ldots,L_n)$ which is a list of instructions. $L_n$ is the instruction `halt`, and for $0 \leq l < n$, $L_i$ is one of:
  - x:= x+1
  - y:= y+1
  - if x = 0 then goto $L_j$ else x := x – 1, $0 \leq j \leq n$
  - if y = 0 then goto $L_j$ else y := y – 1, $0 \leq j \leq n$
- An execution sequence of M is a sequence of states $s_k = (L_{i_k},x,y)$, where $L_{i_k}$ is the current instruction at $s_k$, and x,y are the contents of x and y.
- $s_{k+1}$ is obtained from $s_k$ by executing $L_{i_k}$.
- The initial state $s_0 = (L_{i_0}, m,0) = (L_0, m,0)$ for some m.
- If for some k, $s_k = (L_n,x,y)$, the computation of M has halted and M has computed y = f(m)

# Examples

/* L1 is executed m times and then this program halts*/
$L_0$:if x=0 then goto $L_2$ else x:=x-1
$L_1$:if y=0 then goto $L_0$ else y:=y-1
$L_2$:halt

**Execution where x's initial value=2**

$$s_0\,(L_0,2,0) \rightarrow s_1\,(L_1,1,0) \rightarrow s_2\,(L_0,1,0) \rightarrow s_3\,(L_1,0,0) \rightarrow s_4\,(L_0,0,0) \rightarrow s_5\,(L_2,0,0)$$

*(note that $L_{i_0} = L_0$, $L_{i_1} = L_1$, $L_{i_2} = L_0$, , $L_{i_3} = L_1$, etc)*

/* $L_0$ is executed infinitely, i.e., this program never halts */
$L_0$:x := x + 1
$L_1$:if y=0 then goto $L_0$ else y:=y-1
$L_2$:halt

**Execution where x's initial value=0**

$$s_0\,(L_0,0,0) \rightarrow s_1\,(L_1,1,0) \rightarrow s_2\,(L_0,1,0) \rightarrow s_3\,(L_1,2,0) \rightarrow s_4\,(L_0,2,0) \rightarrow \cdots\cdots$$

# Validity in the predicate calculus

- Thm 5.43 (Church) Validity in the predicate calculus is undecidable
  - *Caution: the proof of Thm 5.43 in the textbook has several flaws…*
- For every two-register machine M, we construct a formula $S_M$ s.t. $S_M$ is valid iff M terminates when started in the state $(L_0,0,0)$:
  - $S_M = (\wedge_{(i=0..n-1)} S_i \wedge p_0(0,0)) \rightarrow \exists z_1 z_2\, p_n(z_1,z_2)$
    - Intuitive meaning of $p_i$ is as follows
      - $v_{\mathcal{I}}(p_i(m',m'')) = T$ iff there exists some state $s_k=(L_i,m',m'')$
  - $S_i$ is defined by cases of the instruction $L_i$

| $L_i$ | $S_i$ |
|---|---|
| `x := x + 1` | $\forall x \forall y (p_i(x,y) \rightarrow p_{i+1}(s(x),y))$ |
| `y := y + 1` | $\forall x \forall y (p_i(x,y) \rightarrow p_{i+1}(x,s(y)))$ |
| `if x = 0`<br>`then goto Lj`<br>`else x := x - 1` | <br>$\forall x (p_i(a,x) \rightarrow p_j(a,x)) \wedge$<br>$\forall x \forall y (p_i(s(x),y) \rightarrow p_{i+1}(x,y)))$ |
| `if y = 0`<br>`then goto Lj`<br>`else y := y - 1` | <br>$\forall x (p_i(x,a) \rightarrow p_j(x,a)) \wedge$<br>$\forall x \forall y (p_i(x,s(y)) \rightarrow p_{i+1}(x,y)))$ |

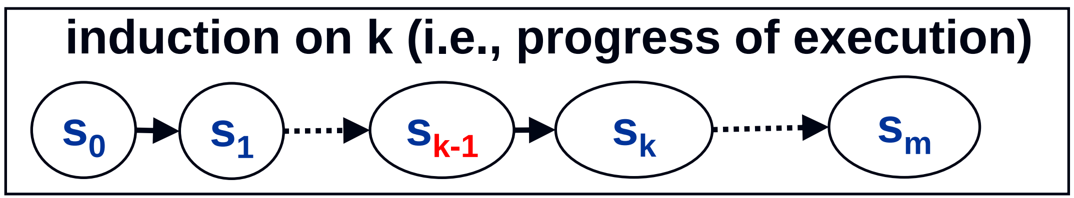
# Example of $S_M$

| | |
|---|---|
| /* y=x+1 for x <= 1* */<br>$L_0$:if x=0 then goto $L_4$ else x=x-1<br>$L_1$:y:=y+1<br>$L_2$:if x=0 then goto $L_4$ else x=x-1<br>$L_3$:y:=y+1<br>$L_4$:halt | $S_M = (p_0(0,0) \wedge$<br>$(\forall x(p_0(0,x) \rightarrow p_4(0,x)) \wedge \forall xy(p_0(s(x),y) \rightarrow p_1(x,y)))) \wedge$<br>$\forall xy\ (p_1(x,y) \rightarrow p_2(x,s(y)) \wedge$<br>$(\forall x(p_2(0,x) \rightarrow p_4(0,x)) \wedge \forall xy(p_2(s(x),y) \rightarrow p_3(x,y)))) \wedge$<br>$\forall xy(p_3(x,y) \rightarrow p_4(x,s(y))\ )$<br>$\rightarrow$<br>$\exists z_1 z_2\ p_4(z_1,z_2)$ |

- Intuitive meaning of $S_M$:
  - Given a two-register machine M,
    - execution of M ($\wedge_{i=0..n-1} S_i \wedge p_0(0,0)$)
    - reaches ($\rightarrow$)
    - the `halt` instruction ($\exists z_1 z_2\ p_n(z_1,z_2)$)

# TRM halts $\rightarrow$ $A_{TRM}$ is valid (1/2)

- Suppose that the execution $s_0,\ldots s_m$ of M halts and let $\mathcal{I}$ be an arbitrary interpretation for $S_M$. If $v_{\mathcal{I}}(S_i) = F$ (for $0 \leq i < n$) or $v_{\mathcal{I}}(p_0(0,0)) = F$, then trivially $v_{\mathcal{I}}(S_M) = T$
- Thus, we assume that $(\wedge_{(i=0..n-1)} S_i \wedge p_0(0,0))$ is true
  - since we need only consider interpretations that satisfy the antecedent of $S_M$
- We show by induction on k that $v_{\mathcal{I}}(\exists z_1 z_2\ p_{i_k}(z_1,z_2)) = T$
  - $p_{i_k}$ is the predicate associated with the label $L_{i_k}$ in state $s_k$
    - Mind the incorrect notation in the textbook where $L_k$ and $p_k$ is used instead of $L_{i_k}$ and $p_{i_k}$
  - For $k=0$, $v_{\mathcal{I}}(\exists z_1 z_2\ p_{i_0}(z_1,z_2)) = v_{\mathcal{I}}(\exists z_1 z_2\ p_0(z_1,z_2)) = T$ since $v_{\mathcal{I}}(p_0(0,0))=T$ from the assumption



# TRM halts $\rightarrow$ $A_{TRM}$ is valid (2/2)

- For $k > 0$, the result follows by induction by cases according to the instruction at $L_{i_{k-1}}$
  - For $x := x+1$ at $L_{i_{k-1}}$:
    - $v_{\mathcal{I}}(\forall xy\ (p_{i_{k-1}}(x,y) \rightarrow p_{i_{k-1}+1}(s(x),y))) = T$ by the assumption
    - $v_{\mathcal{I}}(\exists z_1 z_2\ p_{i_{k-1}}(z_1,z_2)) = T$ by the inductive hypothesis
    - From the above two facts, $v_{\mathcal{I}}(\exists z_1 z_2\ p_{i_{k-1}+1}(s(z_1),z_2)) = T$
    - $v_{\mathcal{I}}(\exists z_1 z_2\ p_{i_{k-1}+1}(s(z_1),z_2)) = v_{\mathcal{I}}(\exists z_1 z_2\ p_{i_k}(s(z_1),z_2)) = T$ since $p_{i_{k-1}+1} = p_{i_k}$
      - We can conclude $v_{\mathcal{I}}(\exists z'_1 z_2\ p_{i_k}(z'_1,z_2)) = T$ since $\exists x\ A(f(x)) \rightarrow \exists x' A(x')$.
    - By induction, this holds for all k.
  - For if $x=0$ then goto $L_j$ else $x=x-1$ at $L_{i_{k-1}}$:
    - …
    - By induction, this holds for all k.
- Since M halts, in the final state $s_m$, $L_{i_m} = L_n$ the halt instruction, so $v_{\mathcal{I}}(\exists z'_1 z_2\ p_n(z'_1,z_2)) = T$ and $v_{\mathcal{I}}(S_M) = T$.
- Since $\mathcal{I}$ was arbitrary, $S_M$ is valid

# TRM halts $\leftarrow$ $A_{TRM}$ is valid

- Suppose that $S_M$ is valid, and consider an interpretation $\mathcal{I}$ s.t.
  - $\mathcal{I} = (\mathcal{N}, \{P_0,\ldots,P_n\}, \{succ\}, \{0\})$ where
    - $(x,y) \in P_i$ iff $(L_i,x,y)$ is reached by the register machine M when started in $(L_0,0,0)$
- We will show that antecedent of $S_M$ is true in $\mathcal{I}$. So, the conclusion of $S_M$ is also true, which means that M reaches the halt instruction since $(x,y) \in P_i$ iff $(L_i,x,y)$ is reached
  - The initial state is $(L_0,0,0)$ so $(0,0) \in P_0$ and $v_{\mathcal{I}}(p_0(0,0)) = T$
- We will show that if the computation has reached $L_i$, then $v_{\mathcal{I}}(S_i)=T$.
  - Assume as an inductive hypothesis that if the computation has reached $L_i$, it has done so in a computation of length–1 in state $s_{k-1}=(L_i,x_i,y_i)$, so $(x_i,y_i)\in P_i$.
  - The proof is by cases on the instruction $L_i$
    - For $L_i$ = x:= x+1, the computation can reach the state $s_k = (L_{i_k}, succ(x_i), y_i) = (L_{i_{k-1}+1}, succ(x_i),y_i)$, so $v_{\mathcal{I}}(S_i) = T$
    - For $L_i$ = if x=0 then goto $L_j$ else x := x -1, …so $v_{\mathcal{I}}(S_i) = T$
- Since $S_M$ is assumed valid, $v_{\mathcal{I}}(\exists z_1 z_2\, p_n(z_1,z_2))=T$ and $v_{\mathcal{I}}(p_n(m_1,m_2))=T$ for some natural numbers $m_1$, $m_2$. Thus M halts and computes $m_2 = f(0)$