ILLUSTRATION: Alloy expression to SAT formula

Boolean variables: Man: m_0, m_1

Woman: w_0, w_1

parents: $p_{00}, p_{01}, p_{10}, p_{11}$

spouse: $s_{00}, s_{01}, s_{10}, s_{11}$

Constraints from signatures: $m_0 \oplus w_0 \;\wedge\; m_1 \oplus w_1$

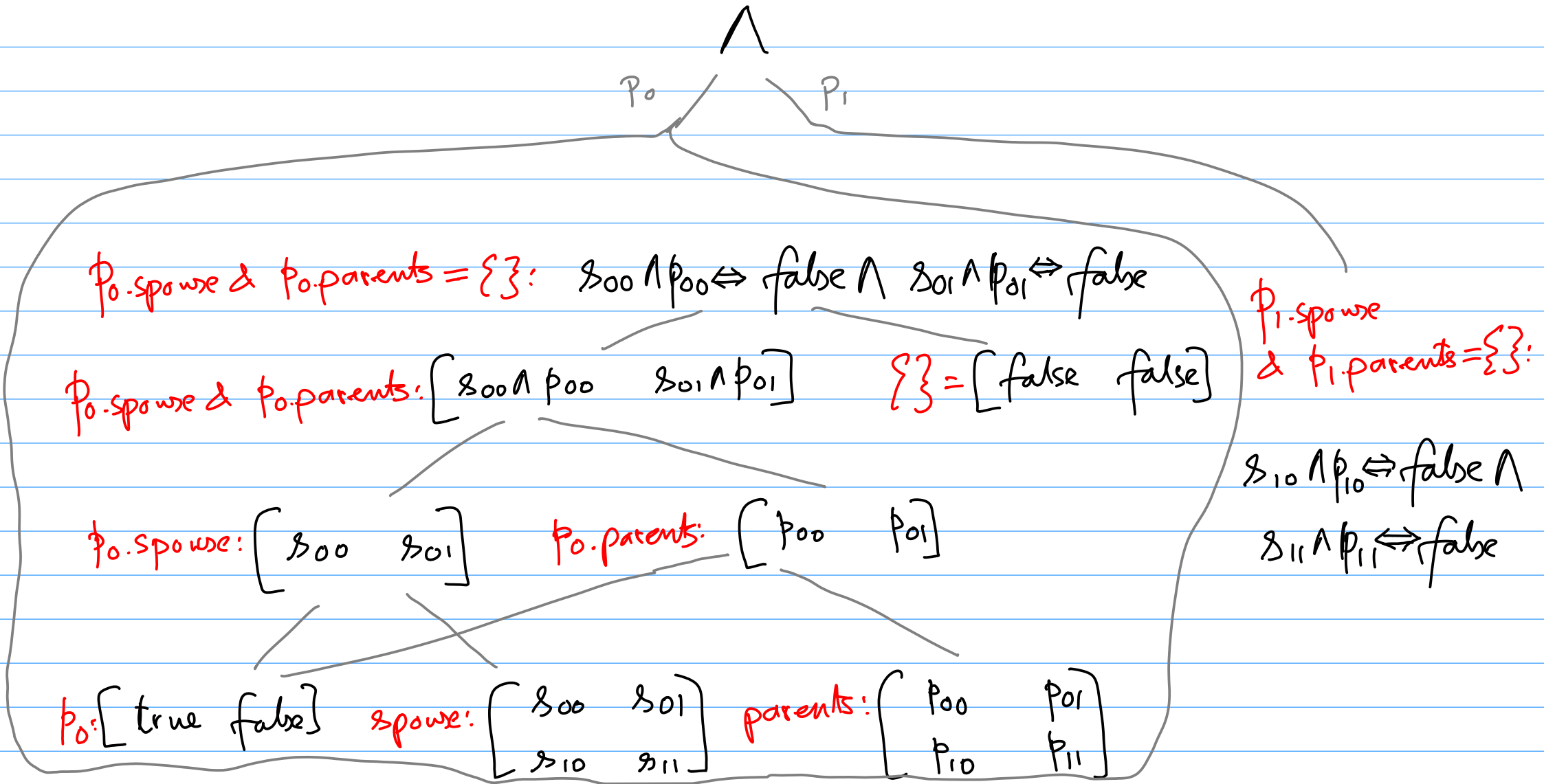
1) Constraint for Alloy expression Man.spouse in Woman:

$$((m_0 \wedge s_{00}) \vee (m_1 \wedge s_{10}) \Rightarrow w_0) \wedge ((m_0 \wedge s_{01}) \vee (m_1 \wedge s_{11}) \Rightarrow w_1)$$

Man.spouse: $\begin{bmatrix} (m_0 \wedge s_{00}) \vee (m_1 \wedge s_{10}) & (m_0 \wedge s_{01}) \vee (m_1 \wedge s_{11}) \end{bmatrix}$ Woman: $\begin{bmatrix} w_0 & w_1 \end{bmatrix}$

Man: $\begin{bmatrix} m_0 & m_1 \end{bmatrix}$ spouse: $\begin{bmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{bmatrix}$

2) (constraint for all p : Person | (p.spouse & p.parents) = {}



ALGORITHM: Alloy expression to SAT formula

Terminology:

Global variable: relation, or non-top-level signature

TSignature: top-level signature

DSignature: derived signature

Variable: Global variable | Quantified variable

R-expr: Alloy relational expression

B-expr: Alloy formula

Formula: Propositional formula

Matrix: Matrix/vector of formulas

n_S: Given bound for top-level signature S

Type declarations:

Env: Variable → Type

Type: TSignature | TSignature × TSignature | DSignature

// Assume all relations are
// binary relations, from
// top level sigs to top
// level sigs.

Procedures

Formula main(B-expr exp) {

env = a mapping where each non-top-level signature is mapped to its corresponding top-level signature, & each global relation γ is mapped to a term (S_1, S_2) where S_1 is the top-level signature corresponding to γ's domain & S_2 is the top-level signature corresponding to γ's range.

return Translate-B-expr(exp, env);

}

Matrix Translate-R-Expr(R-expr exp, Env env) {

case exp is r, where r is a relation:

$(S_1, S_2) = env(r)$

m = a new matrix of size $n_{S_1} \times n_{S_2}$

for all $0 \le i \le n_{S_1}$, $0 \le j \le n_{S_2}$

$m[i,j] = r_{ij}$, where r_{ij} is the (i,j)th boolean variable corresponding to r

return m;

case exp is v, where v is a non-top-level signature:

$S = env(v)$

return $[v_0 \; v_1 \; \cdots \; v_{n_S - 1}]$

case exp is a top-level signature S:

return [true true .... true] // size of vector is n_S

case exp is w_i, an instance of a quantified variable w:

v = env (w). S = top-level sig on which v is based.

m = a new row vector of size n_S

if $(S = v)$ $m[i]$ = true else $m[i] = v_i$ // v_i is i^{th} boolean // corresponding to

for all j in $0 \cdots (n_S - 1)$, $j \neq i$: $m[j]$ = false // The signature v

return m;

case exp is $e_1 \cdot e_2$:

m_1 = Translate-R-expr (e_1, env);

m_2 = Translate-R-expr (e_2, env);

assert (# cols in m_1 = # rows in m_2);

$m_3 = m_1 \times m_2$ // use '&&' instead of '*' & '||'
// instead of '+'

return m_3;

.... similarly, handle all other relational operators ....

}

Formula Translate-B-expr(B-expr exp, Env env) {

case exp is "all v:T | f":

let S be top-level signature corresponding to T

for i in $0 \dots (n_S - 1)$ {

f_i = f, with all occurrences of v replaced with v_i

res_i = Translate-B-expr(f_i, env ++ ($v_i \to T$))

}

return $\bigwedge_{0 \dots n_S} res_i$;

case exp is "e_1 in e_2":

m_1 = Translate-R-expr(e_1, env);

m_2 = Translate-R-expr(e_2, env);

assert (dimensions of m_1 = dimensions of m_2);

p = num_rows(m_1); q = num_cols(m_1);

return $\bigwedge_{\substack{0 \le i < p, \\ 0 \le j < q}} (m_1[i,j] \Rightarrow m_2[i,j])$

case exp is "some e_1":

m = Translate-R-expr(e_1, env);

p = num_rows(m); q = num_cols(m);

return $\bigvee_{\substack{0 \le i < p, \\ 0 \le j < q}} m[i,j]$

... similarly, all other cases of boolean operators...

}