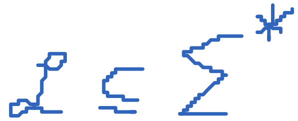
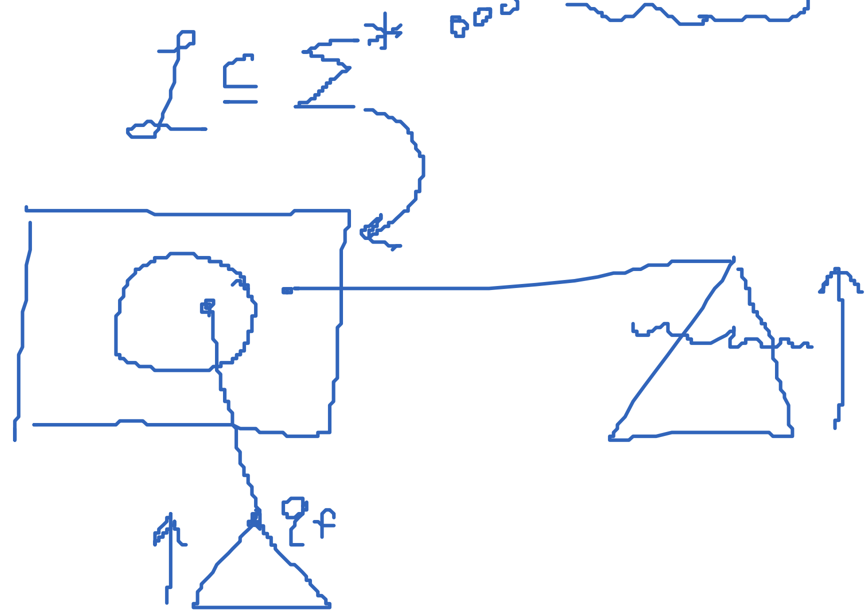
Bottom-up rewrite systems (BURS)

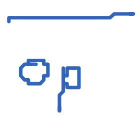
Thursday, March 12, 2020 11:33 AM

String automata →Tree automata





$$Times\big(q_1(X_1), q_5(X_2)\big) \rightarrow q_{17}(Times(X_1, X_2))$$



≈ Rebuilding

Suggests: Why not build something new?

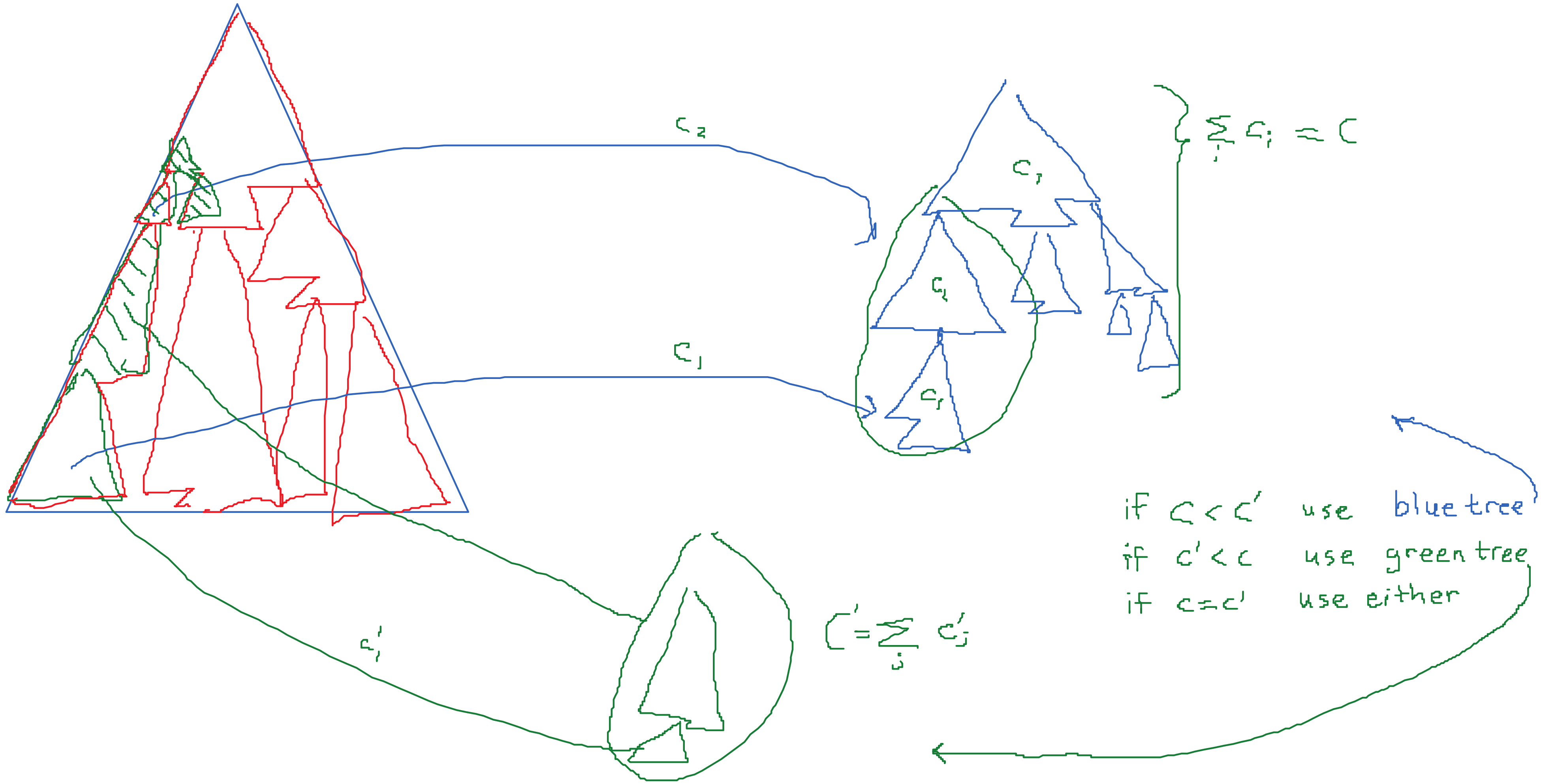
$$Times(q_{nz}(X_1), q_z(X_2)) \rightarrow q_z(0)$$

L_1 (e.g., IR language)

L_2 (e.g., Code language)

Dynamic programming (1)

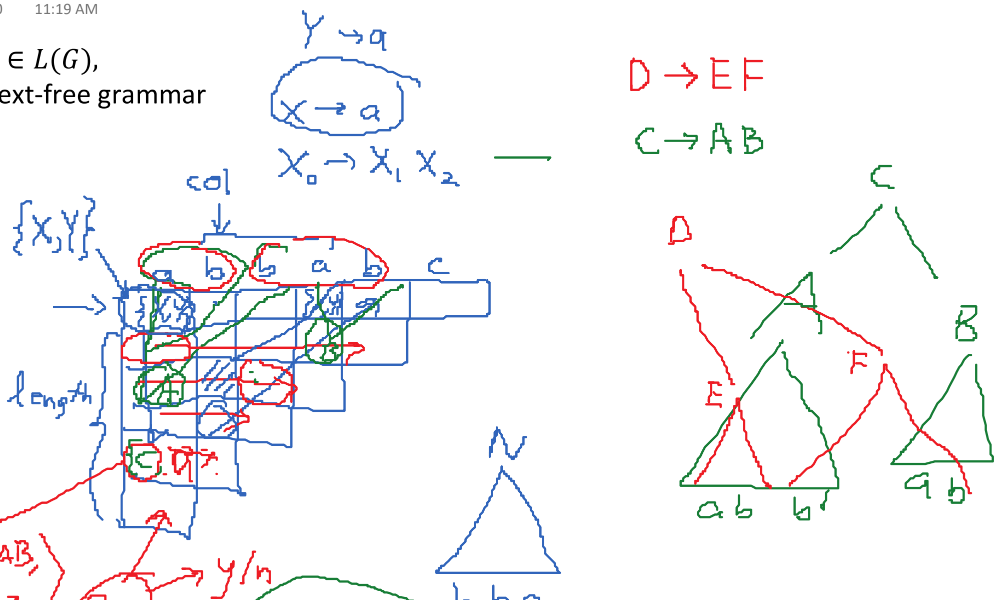
Friday, March 13, 2020 11:11 AM

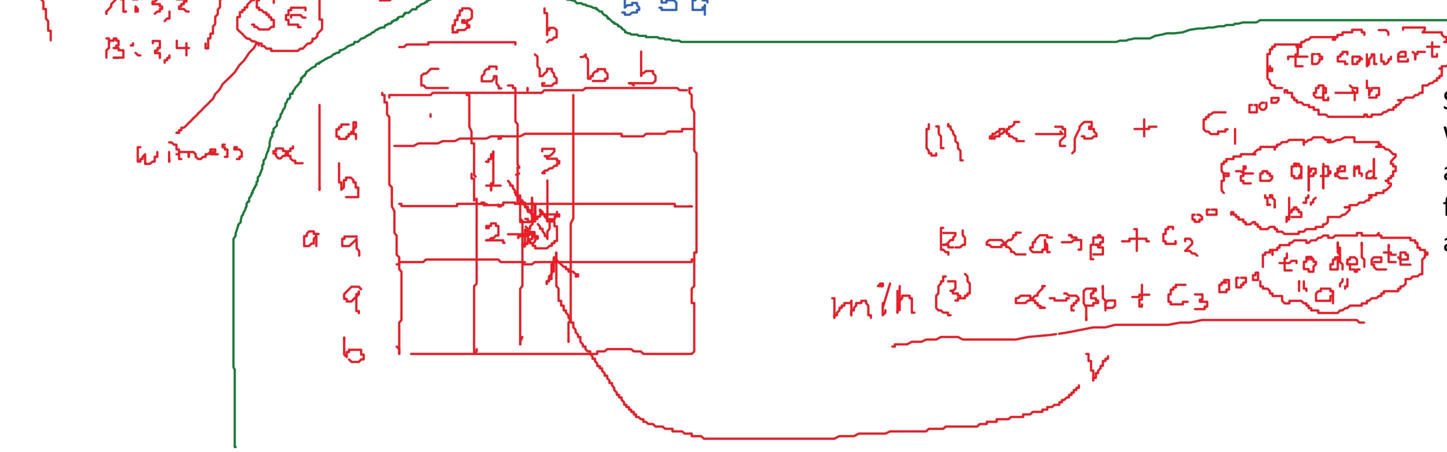


Dynamic programming (2)

Friday, March 13, 2020 11:19 AM

CYK Algorithm: $\omega \in L(G)$,
where G is a context-free grammar





String-to-string correction:
What is the cost of converting, e.g.,
abaab into cabbb (given costs c1, c2, and c3
for converting one letter to another, appending
a letter, and deleting a letter, respectively)

BURS for Code Generation

Thursday, March 12, 2020 10:59 PM

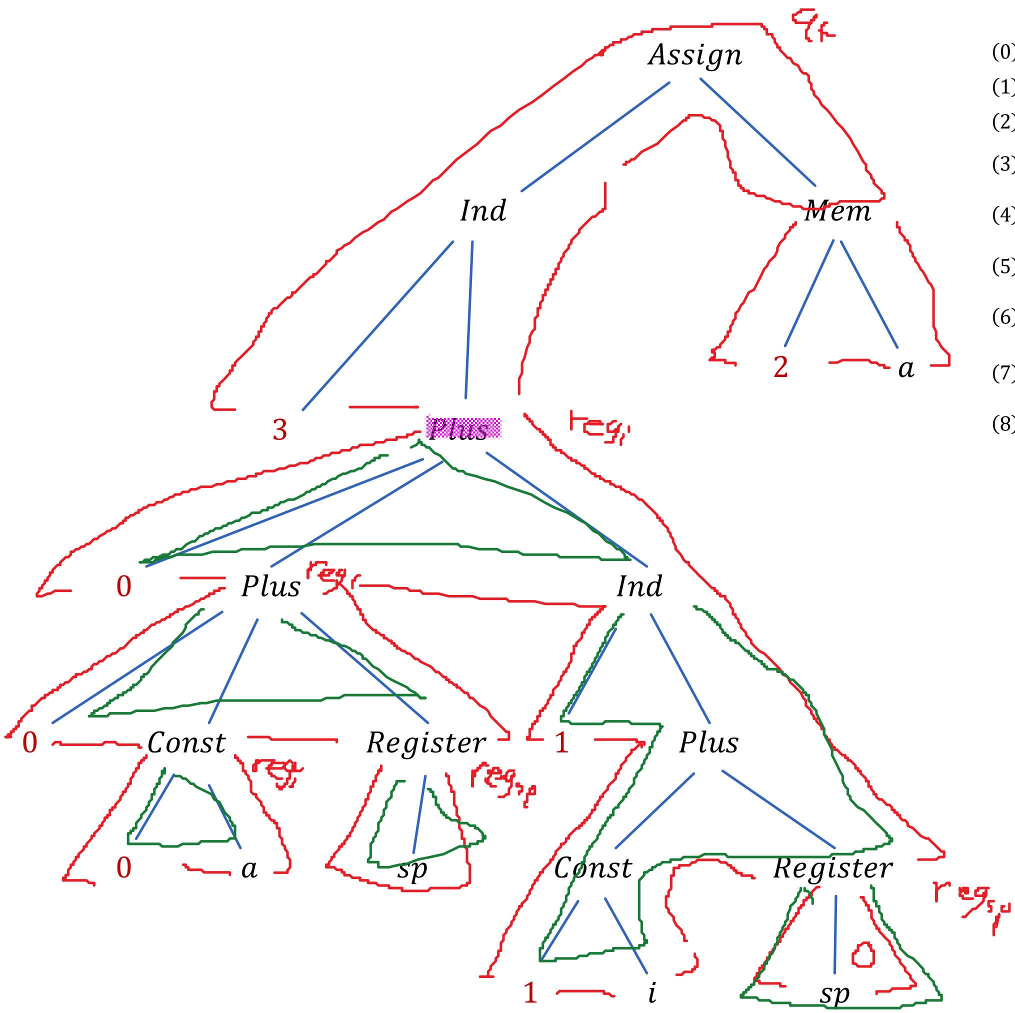
$\Sigma_{IR} = \{Assign^2, Mem^2, Const^2, Plus^3, Ind^2, Register^1, \dots, constants^0, \dots addresses^0\}$
$\Sigma_{Code} = \{NoOp^0, R^1, Imm^1, Indirect^1, Mov^2, Cons^2, \dots, constants^0, addresses^0, \dots\}$
$Q = \{reg_i, q_f\}$

In language $Code$, $Add(R(i), R(j))$ adds $R(i)$ to $R(j)$, and leaves the result in register $R(i)$

(0) $Register(i) \xrightarrow{[0]} reg_i(NoOp(\,))$
(1) $Const(i,c) \xrightarrow{[2]} reg_i(Mov(R(i), Imm(c)))$
(2) $Mem(i,a) \xrightarrow{[2]} reg_i\left(Mov(R(i), Indirect(a))\right)$
(3) $Assign(Mem(*,a), reg_i) \xrightarrow{[2]} q_f\left(Cons\left(reg_i, Mov(Indirect(a,0), R(i))\right)\right)$
(4) $Assign(Ind(*,reg_i), reg_j) \xrightarrow{[2]} q_f\left(Cons\left(reg_i, Cons\left(reg_j, Mov(Indirect(R(i),0), R(j))\right)\right)\right)$
(5) $Ind(i, Plus(*, Const(*,c), reg_j)) \xrightarrow{[2]} reg_i\left(Cons\left(reg_j, Mov(R(i), Indirect(R(j),c))\right)\right)$
(6) $Plus(i, reg_i, Ind(*, Plus(*, Const(*,c), reg_j))) \xrightarrow{[2]} reg_i\left(Cons\left(reg_i, Cons\left(reg_j, Add(R(i), Indirect(R(j),c))\right)\right)\right)$
(7) $Plus(i, reg_i, reg_j) \xrightarrow{[1]} reg_i\left(Cons\left(reg_i, Cons\left(reg_j, Add(R(i), R(j))\right)\right)\right)$
(8) $Plus(i, reg_i, Const(1)) \xrightarrow{[1]} reg_i\left(Cons\left(reg_i, Incr(R(i))\right)\right)$

Example from Twig paper (adapted)

Friday, March 13, 2020 12:13 AM



$(0)\ Register(i) \xrightarrow{[0]} reg_i(NoOp(\,))$

$(1)\ Const(i,c) \xrightarrow{[2]} reg_i(Mov(R(i), Imm(c))$

$(2)\ Mem(i,a) \xrightarrow{[2]} reg_i\left(Mov(R(i), Indirect(a))\right)$

$(3)\ Assign(Mem(*,a), reg_i) \xrightarrow{[2]} q_f\left(Cons\left(reg_i, Mov(Indirect(a,0), R(i))\right)\right)$

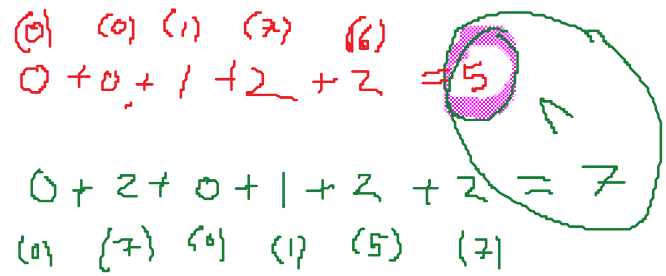
$(4)\ Assign(Ind(*,reg_i), reg_j) \xrightarrow{[2]} q_f\left(Cons\left(reg_i, Cons\left(reg_j, Mov(Indirect(R(i),0), R(j))\right)\right)\right)$

$(5)\ Ind(i, Plus(*, Const(*,c), reg_j)) \xrightarrow{[2]} reg_i\left(Cons\left(reg_j, Mov(R(i), Indirect(R(j),c))\right)\right)$

$(6)\ Plus(i, reg_i, Ind(*, Plus(*, Const(*,c), reg_j))) \xrightarrow{[2]} reg_i\left(Cons\left(reg_i, Cons\left(reg_j, Add(R(i), Indirect(R(j),c))\right)\right)\right)$

$(7)\ Plus(i, reg_i, reg_j) \xrightarrow{[1]} reg_i\left(Cons\left(reg_i, Cons\left(reg_j, Add(R(i), R(j))\right)\right)\right)$

$(8)\ Plus(i, reg_i, Const(1)) \xrightarrow{[1]} reg_i\left(Cons\left(reg_i, Incr(R(i))\right)\right)$



Normalization

Friday, March 13, 2020 11:45 AM

Multi-level patterns

$Baz(Goo(q_2, q_3), q_4) \xrightarrow{[w_1]} q_1(\dots)$

$Baz(q_5, Gaz(q_6)) \xrightarrow{[w_2]} q_9(\dots)$

Single-level patterns

$Goo(q_2, q_3) \xrightarrow{[0]} q_a(\dots)$ // q_a is a new "intermediate state"; the weight of such a rule is 0

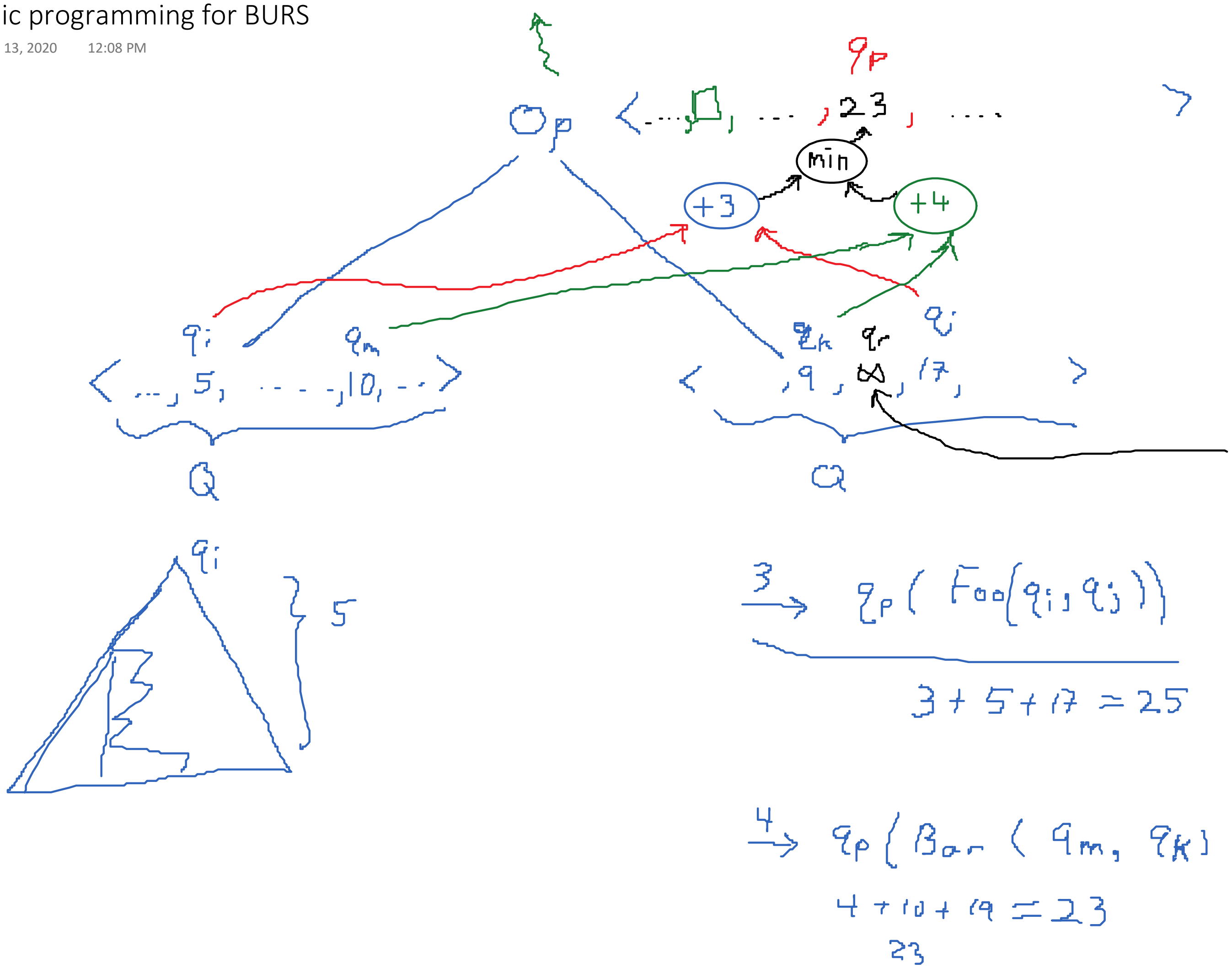
$Baz(q_a, q_4) \xrightarrow{[w_1]} q_1(\dots)$ // weight incurred only when topmost operator of original pattern reached

$Gaz(q_6) \xrightarrow{[0]} q_b(\dots)$

$Baz(q_5, q_b) \xrightarrow{[w_2]} q_9(\dots)$

Dynamic programming for BURS

Friday, March 13, 2020 12:08 PM



∞ means no tiling of the subtree exits that leads to a tree in state q_r

Last Thoughts

Friday, March 13, 2020   12:35 PM

String-automata/languages → Tree-automata/languages
The right conceptual model for BURS:
  "A BURS is a weighted tree transducer, where weights are totally ordered (e.g., from $\mathbb{N} \cup \{\infty\}$)
Can test desirable properties, e.g., does the BURS handle all IR trees?
  language containment: does $\mathrm{L}(\mathrm{IR}) \subseteq \mathrm{L}(\mathrm{Pat})$ hold?