1F-2: Even this is an "old" paper, I think it still applies to contemporary PL designs. I agree with the author that "the magnitude of the tasks we wish to computers to perform is growing faster than the cost-effectiveness of the hardware". Nowadays, we have more powerful hardware thanks to the "Moore's Law", but the tasks we explore also become growing complex, like quantum computing, artificial intelligence". Also, the size of data we are processing nowadays is almost inconceivable at the time lots of today widely used PL were designed. So, making the object code efficient is always desireable for a long-term running PL.

I somewhat disagree with his arguement that it is "irritating" when someone says that future hardware designs should be oriented towards the implementation of this complexity. As he said simplicity is needed, but if any PL is popular enough to drive the orientation of hardware design, it is also acceptable. I think the development of GPU would not be that fast as in recent years without the incredible growing demands from deep learning community. It could be an enlightenment for hardware designers if some PLs or programing framework can lead their orientation.

IF-3 $\dfrac{\langle e_1,\sigma\rangle \Downarrow n_1 \quad \langle e_2,\sigma\rangle \Downarrow n_2 \quad \langle n_3 * n_2 \le n_1,\sigma\rangle \Downarrow \text{true} \quad \langle n_1 - n_2 \le n_3 \cdot n_2,\sigma\rangle \Downarrow \text{true} \quad \langle n_1 - n_2 = n_3 \cdot n_2,\sigma\rangle \Downarrow \text{false}}{\langle e_1 / e_2,\sigma\rangle \Downarrow n_3}$

We want $n_1 / n_2 = \left\lfloor \frac{n_1}{n_2} \right\rfloor$ arithmetically.

Assume $n_3 = \left\lfloor \frac{n_1}{n_2} \right\rfloor$, then $n_3 \cdot n_2 \le n_1$ and $n_1 - n_2 < n_3 \cdot n_2$

I don't check if n_2 is 0 since if it is zero then the premise never holds.

IF-4 $\dfrac{\langle e,\sigma\rangle \Downarrow n_1 \quad \langle c,\sigma[x := n_1]\rangle \Downarrow \sigma'}{\langle \text{let } x = e \text{ in } c,\sigma\rangle \Downarrow \sigma'[x := \sigma(x)]}$

Change x back to the original value after command c.

IF-5 rules $\langle \text{let } x = e \text{ in } c,\sigma\rangle \longrightarrow \langle x := e;\ c;\ x := \sigma(x),\sigma\rangle$

redex let $x = e$ in c

Context: $H ::= \text{let } x = e \text{ in } H$

let can be transferred into three semicolon-separated commands: First set x to e and execute c and change x back to the original value. Then the redex is "let $x = e$ in c". The context extends to let itself since let can be nested.

1 HW1 (select all pages: your first page has your name and bookkeeping, and all others are anonymous))

- **0 pts** Correct