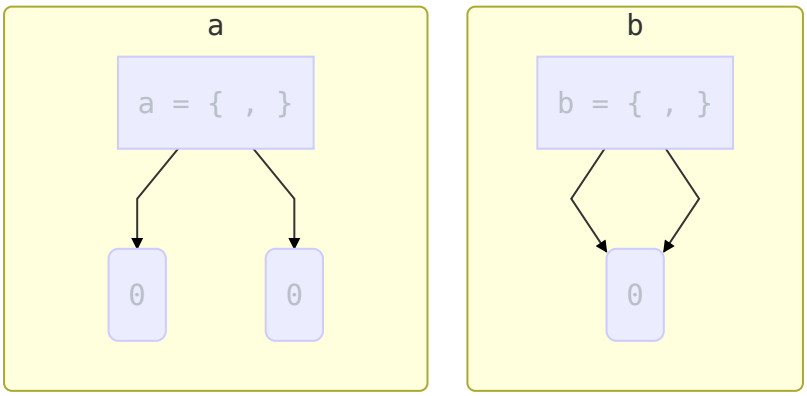
第 13 章 参照

> テキストの解答要約はこんな感じで引用表現にする（引用じゃないけど）

演習 13.1.1. [⋆]



演習 13.1.2. [⋆⋆]

- 観察：let で deref している部分が失われている。

同じ動作をしない。

assign 式の右辺の else 節で deref しているが、これが評価されるのは破壊的代入の後になる。つまり、ある a, m, v, に対して update a m v を呼び出すと、a が即座に右辺の形に破壊的代入される。その後 lookup などで n (not equal m) を与えて、E-IfElse で評価されたときに初めて、a が deref される。そしてこの a は破壊的代入の後の a そのものである。恐ろしいことに else 節では deref した a に n を適用しているので、この評価は停止しない。

> 正確に言うと、停止しないというか、発散する。

演習 13.1.3. [⋆⋆]

- ぶら下がり参照: dangling reference
- 型安全 = 進行 + 保存
- ぶら下がり参照の文章、なんかへんてこだけど、多分これを具体的に言えってことだと思う

保存則が脅かされる。（詳細は諦めた）

> free プリミティブを使うと、行き詰まり状態が作れる。（進行則が脅かされる）ポイントは、あるセルに対する別名参照を作ったとき、元の参照を free しても別名参照に対しては操作ができてしまうこと。

Note: 13.3

- 参照の定式化では、「位置」は距離や算術演算を持たない、単なる集合の要素として考える
  - これによってポインタ算術をあり得ないものとし、型に関する議論をシンプルにできる
  - C言語では「位置」はポインタであり、バイト列上の添字の数値で、比較や算術演算ができ、型によってセルは異なるサイズを持つ
- 評価規則の拡張
  - 関数適用は副作用を起こさない
  - ストアにおける位置はプログラマは直接触らないが、評価規則上には現れる**中間言語**的要素
    - E-Derefで値（位置）まで簡約→E-DerefLocでストア上のメタ変数μから値を取り出す
    - E-Assign1, 2で l := v の形まで簡約→E-Assignでunitに評価、μ上のlをvにマップする（更新）
    - E-Refで値vまで簡約→E-RefVで値vを位置lに簡約し、μをl→vを含むように拡張
- ではガベージコレクションはどうする？

演習 13.3.1. [⋆ ⋆ ⋆]

方針：スコープを導入する。スコープに入る際にストアのメタ変数 μ を中間言語に退避する (scope_μ のように)。スコープ内の処理でメタ変数が変化していくことは許容して、スコープを出る際に元のメタ変数に戻す。

$$\mathsf{scope}\ t_1 \,|\mu \to \mathsf{scope}_\mu\ t_{11} \,|\mu \qquad (\text{E-ScopeIn})$$

$$\frac{t_1 \,|\mu \to t_1' \,|\mu'}{\mathsf{scope}_\nu\ t_1 \,|\mu \to \mathsf{scope}_\nu\ t_1' \,|\mu'} \qquad (\text{E-Scope})$$

$$\mathsf{scope}_\nu\ v_1 \,|\mu \to v_1 \,|\nu \qquad (\text{E-ScopeOut})$$

- メタ変数を中間言語へ退避するのではなく、あらたな構文の状態を拡張して、メタ変数そのものではなくメタ変数のスタックを状態とする、という方法も考えられそう。

> - 参照が連鎖することをちゃんと考えよう（reachableの定義）
> - GCが任意の1ステップ評価の直後に発生し得るように、1ステップ評価を拡張しよう。
> - (4)のこの定義で「最も外側のレベル」でGCが走ることを保証できるってのがいまいちピンと来ない…