PoPL Lecture 7

Zubair Abid

2020-09-01

In class

- We know the **Well-foundedness Property of Inductive Terms**: There is no inductive term `T` s.t. `T` is a proper subterm of itself:
  - So we can't have terms like `T = g(T)`
  - But we might a term like that for *streams*

    ```
    g←     Circular terms
    ```

  - The following is seemingly an aside. Skip if you need to.
    * Consider the following definition

      ```
      TREE              DAG
        f                 f
       / \               ( )
      a   a               a
      -----             -----
      f a a             f a a
      ```

      In the DAG, the 'a' is shared. It is like

      ```
      (define (a) (list 'a))
      (define (f t1 t2) (list 'f t1 t2))
      (let ([t (a)])
        (f t t))
      ```

      But in the tree form, it is more

      ```
      (let ([t1 (a)]
            [t2 (a)])
        (f t1 t2))
      ;; note that t1 and t2 are different
      > (eq? (a) (a))
      #f
      > (equal? (a) (a))
      #t
      ```

Term Graphs

And defining them in terms of equations

- **Term Graphs**: A term graph over a signature Σ is a structure <V, h, →> where
  1. **V** is a set of vertices
  2. **h**: V $\to$ Σ (head)
  3. $\to$ V x N x V (v,i,v' written as v-i->v')
  4. if h(v) Σ , then v has exactly n out-edges. For each i: 1 i n, there is exactly , then v has exactly 1 out-edge labelled i.
- Defining some *notation*:
  - if h(v) = f and
  - f Σ and
  - v -i-> v_i for 1 i n
  - Then, we write this as

    v f(v … v_n) *(is actually '=' with a dot on top)*
- Consider examples:

```
     (f)t      v = {t  t  t }   t   a()
    / \       h = { t  → f     t   b()
t (a) (b)t          t  → a     t   f(t  t )
G=<v,h,→>            t  → b }
                               this is a term
                               graph written down
                               as a set of equations

f has arity 2, so it needs 2 out-edges

We can also have
    (g)←
```

  For another example, we can see:

Path/Position

- From the term graph above, we can write something like t $^2\to$ t $^1\to$ t
- A simpler way to do that is t $^{21}\to$ t
- This ^{21} is called the **path/position**
- The path 21 takes t to t

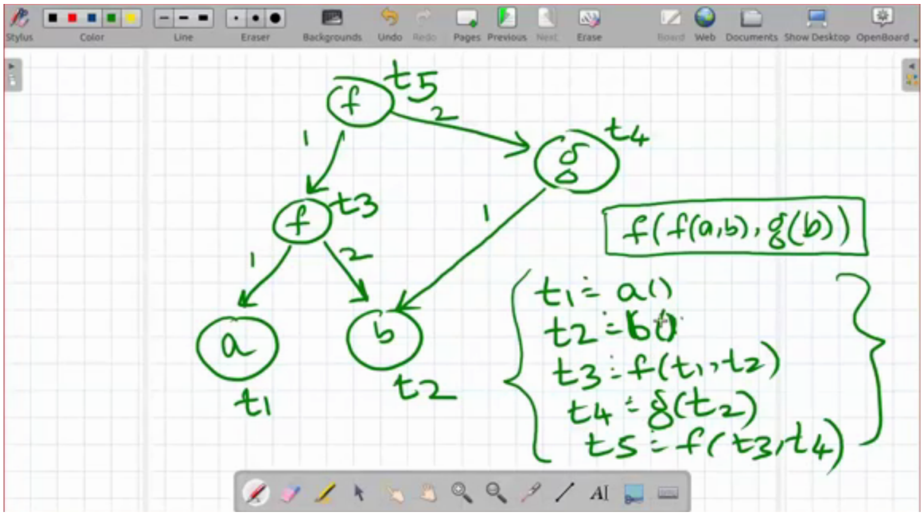

Figure 1: More complex term graph as equations

- **REF rule**:

  `v  []-> v`

  TRANS rule:

  `v'  [p]→ v''` and `v  [i]→ v'`, then

  `v  [i.p]→ v''`

- Now, also:

  if `v  [p]→ r` and `v  [p]→ s`, then `r == s`

- TODO after this
- Position space
- paths == position
- pos(t1): all paths to all accessible vs.
- Term can be reconstructed from pos v and $\rightarrow$
- Position space: set of positions p, prefix closed: for all terms of the set, each prefix is also a term in the set. *Motivation*: Eliminating v

  Basically defines term v, without the v. 1 Position space defines a term graph's subtree.

- Behaviour: mapping from terms to , but upwards complete. Interaction.
- term over Σ is a behavious over Σ

Behaviour