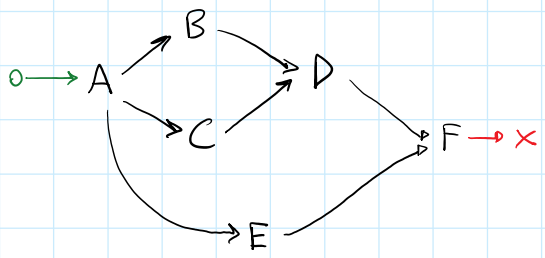
Consider the following DAG:



We can build a map of dependencies in two steps.

1) List the dependencies:

F ← [D, E]
E ← [A]
D ← [B, C]
C ← [A]
B ← [A]
A ← ∅

2) Build a map using the dependencies as keys, grouping the dependent nodes:

when_all(...).then(...)

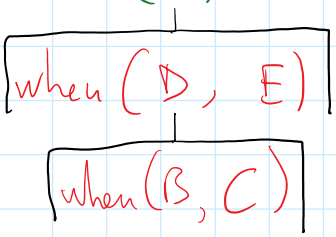
[F]:	[X]	→ len(k) == len(v)	→ linear	→ k.then(v)
[D, E]:	[F]	→ len(k) > len(v)	→ join	→ when_all(k...).then(v)
[B, C]:	[D]	→ len(k) > len(v)	→ join	→ when_all(k...).then(v)
[A]:	[E, C, B]	→ len(k) < len(v)	→ fork	→ k.then(v...)
[0]:	[A]	→ len(k) == len(v)	→ linear	→ k.then(v)

Maybe the above map could be used to generate this?

∅.then(A.then(

∅.then(A.then(

end.then(F)



))