1 Modelli di Kripke

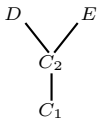
1.1 L'intuizione (molto informale)

Possiamo pensare all'albero delle possibili esercuzioni di un programma. Ogni nodo rappresenta uno stadio dell'esecuzione. Abbiamo diversi rami quando sono possibili diverse evoluzioni nell'esecuzione (a seconda degli input).

Per esempio, immaginiamo che D e E siano comandi di una libreria, C_1 e C_2 i comandi che appaiono nel seguente programm-ino:

- C_1: read *input*;
- C_2: if *input* then D else E.

Questo programma ha un albero di esercuzione di questo tipo:



Usando formule, possiamo esprimere proprieta' del programma, e verificare se sono vere durante tutta l'esecuzione, oppure a partire da un certo stadio. Per esempio, possiamo porre "p: l'operazione D e' stata eseguita"

Quindi abbiamo $V(p)$: l'insieme dei nodi in cui p vale, cioe' l'insieme dei nodi in cui sappiamo che l'operazione D e' stata eseguita.

Per esempio, potremmo voler verificare:
E' vero che in qualunque momento dell'esecuzione del programma iniziale, D e' stato eseguito? No.
E' vero che in qualunque momento di tutte le esecuzioni del programma, siamo sicuri che C_1 e' stato eseguito? Si.
Ci possiamo anche domandare se e' vero che un certo comando non sara' mai eseguito.

Esprimere proprieta' con formule. Consideriamo un albero di esecuzione. Chiamiamo $x, x_1, x_2, ...$ i nodi dell'albero (ogni nodo e' un momento dell'esecuzione).

Scriviamo $x \in V(F)$ per dire *"al nodo x vale F"*

p vale in x se:	da x in poi, p vale (es: "il comando C e' stato eseguito");
$\neg A$ vale in x se:	da x in poi, A non vale mai;
$A\&B$ vale in x se:	da x in poi, A vale, e B vale;
$A \vee B$ vale in x se :	da x in poi, A vale, oppure B vale;
$A \rightarrow B$ vale in x se:	da x in poi, se A vale, allora anche B vale.
	Cioe', da x in poi abbiamo una delle due: – A non vale mai, oppure – Se A vale, anche B vale

Esempio...
p : *il comando C e' stato eseguito*
p : *il comando D e' stato eseguito*

Ogni formula esprime una proprieta':

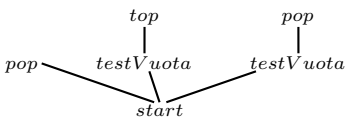
$x \in V(p)$:	da x in poi, il comando C e' stato eseguito;
$x \in \neg p$:	da x in poi, C non sara' mai eseguito;
$x \in V(p\&q)$:	da x in poi, C e' stato eseguito, e D e' stato eseguito;
$x \in V(p \vee q)$:	da x in poi, C e' stato eseguito, oppure D e' stato eseguito;
$x \in V(p \rightarrow q)$:	da x in poi, se C e' stato eseguito , allora anche D. Cioe', da x in poi abbiamo una delle due: • C non e' stato eseguito, oppure • Se C e' stato eseguito, e' stato eseguito anche D.

Per esempio , supponiamo di voler essere sicuri che ogni volta che si esegue pop da una pila, viene eseguito anche il test per verificare che la pila non e' vuota. Potremmo scrivere una formula sul genere di:

eseguito pop(pila) $\rightarrow$ *eseguito testVuota(pila)*

In che stati l'esecuzione di un programma e' sicura? Negli stati in cui e' vero che "Ogni volta che eseguo pop, ho eseguito anche il test", cioe' nei nodi in cui vale *eseguito pop(pila)* $\rightarrow$ *eseguito testVuota(pila)*

Il programma sotto e' sicuro? Dove l'esecuzione e' sicura?



1.2 Un algoritmo

Concretamente, possiamo pensare di annotare ogni nodo dell'albero con le proprieta' (le formule) che sono valide a quello stadio dell'esecuzione: pensiamo di dare ad ogni nodo dell'albero un insieme di etichette.

- Se un nodo ha etichetta A, allora anche tutti i nodi che vengono dopo di lui hanno etichetta A.
- V_{atm} dice per ogni proposizione atomica, quali nodi etichetta.

Scriviamo $x \in V(A)$ per:

"il nodo x ha etichetta A" o anche *"A vale al nodo x"*.

V(A) e' quindi l'insieme dei nodi che hanno etichetta A, cioe' l'insieme dei nodi dove A vale.

Fissato V_{atm} possiamo dire quali nodi hanno etichetta $\neg p, p \vee \neg q, p \rightarrow q$, eccetera...

Il nodo x ha etichetta F se...

$x \in V(P)$**:** da quel momento in poi, ogni nodo ha etichetta P;

$x \in V(A\&B)$**:** da x in poi, ogni nodo ha etichetta A e ha etichetta B.

$x \in V(A \vee B)$**:** da quel momento in poi, ogni nodo ha etichetta A oppure ha etichetta B.

$x \in \neg A$: da quel momento in poi, *nessun nodo* ha etichetta A

$x \in V(A \rightarrow B)$**:** da quel momento in poi, per *ogni nodo y*, se ha etichetta A, allora ha anche etichetta B. Cioe': y non ha etichetta A, o se y ha etichetta A, ha anche etichetta B.

Importante Ben distinguere tra questi tre diverse nozioni:

1. x ha etichetta A;
2. x non ha etichetta A;
3. x ha etichetta $\neg A$ (quando **da x in poi,** *nessun nodo ha etichetta A*).

1.3 Riassumendo

$V(p)$**:** tutti i nodi che hanno etichetta atomica p ($V(p) = V_{atm}(p)$)

$V(A\&B)$**:** $V(A) \cap V(B)$;

$V(A \vee B)$**:** $V(A) \cup V(B)$;

$V(\neg A)$**:** {x: da quel nodo in poi, non ho MAI etichetta A }

$V(A \to B)$**:** {x: da x in poi, ogni volta che un nodo ha etichetta A, ha anche etichetta B.

Cioe' da x in poi abbiamo una delle due, per ***ogni nodo*** y:

- y non ha etichetta A
- se y ha etichetta A, ha anche etichetta B.

1.4 Da sapere

Gli alberi che abbiamo descritto si chiamano modelli di Kripke.

Le cose importanti da ricordare sono:

- *Una formula e' derivabile in LJ se e solo se e' valida in ogni albero* (teorema di completezza).
- Quindi:
  - se trovo un albero T ed una valutazione V_{atm} per cui la formula A non e' valida in T, allora la formula A non e' derivabile.
  - se una formula A e' derivabile, allora per ogni dato albero T, qualunque valutazione prendiamo, A e' sempre valida in T.

Possiamo usare i modelli di Kripke come usiamo le tavole di verita' per LK.

In particolare, li usiamo per dire che una formula non e' derivabile: se costruiamo un **contromodello** (cioe' una valutazione che rende la formula falsa), sappiamo che la formula non e' derivabile.