Pagina	*Riferimento*	*Correzione*
12	Ventiseisimo rigo	if (stato == 2 || stato == 3) → if (stato == 1 || stato == 3)
12	Nono rigo dal basso	[4][2] → [5][3]
13	Quinto rigo	if (stato == 2 || stato == 3) → if (stato == 1 || stato == 3)
24	Listato	Togliere la prima "wait_interrupt"
41	Quindicesimo rigo	port control register → port direction register
45	Eq. (3.1)	$x_{n-i} \rightarrow x_{k-i}$
69	Eq. (5.6)	$dR(t)/dt = Z(t)R(t) \rightarrow dR(t)/dt = -Z(t)R(t)$
78	Eq. (5.31) e Eq. (5.32)	$R_i \rightarrow R$ e $Q_i \rightarrow Q$
79	Eq. (5.34)	$R_{2\ su\ 3}(t)R_v(t)(3e^{-2\lambda t}-2e^{-3\lambda t}) \rightarrow R_{2\ su\ 3}(t) = R_v(t)\ (3e^{-2\lambda t}-2e^{-3\lambda t})$
81	Eq. (5.39)	Prob{funzionante in [0,t]} → Prob {guasto in [0,t]}
83	Secondo rigo di (5.41)	$P_2 \rightarrow P_2(t)$
93	Tab. 6.5	Tutte le righe devono essere del tipo: $10^{-(i+1)} \leq THR < 10^{-i}$
103	Dodicesimo rigo	x mod 9 + y mod 9 = s mod 9 → (x mod 9 + y mod 9) mod 9 = s mod 9
118	Terzultimo rigo	$1+x+x^6 \rightarrow 1+x^2+x^6$
121	Primo rigo	moltiplicazione → addizione
121	Quartultimo rigo	$G(x) \rightarrow G_z(x)$
123	Ottavo rigo dal basso	$g(x) \rightarrow G(x)$
123	Ultimo rigo	due occorrenze: 128 → 127
124	Quattordicesimo rigo	del grado di *G(x)* --> del grado *r-1* di *G(x)*
126	Nono rigo	*x* zero → *p* zero
130	Nono rigo	100 → 010
130	Decimo rigo	1101100 → 1101010 e 1001100 → 1001010
131	Secondo rigo	$2^k \geq k+z+1 \rightarrow 2^z \geq k+z+1$
142	Tab. 9.6	L'ultima riga della tabella deve essere cancellata
153	Terzo rigo	Eecovery → Recovery

169	Quinto punto	*abort* due sensi → *abort* fluisce nei due sensi
181	Tredicesimo rigo dal basso	Il terzo clock --> Il primo clock
205	Sedicesimo rigo	valga passato(T,P) → valga successo(P,T)
231	Sesto rigo dal basso	p → *p*
232	Pagina intera	Pagina corretta allegata
233	Pagina intera	Pagina corretta allegata
235	Righe 5, 6	Sostituire con: • $\sigma, w \models\sim \varphi \iff \sigma, w \not\models \varphi$ • $\sigma, w \models \varphi_1 \wedge \varphi_2 \iff (\sigma, w \models \varphi_1) \wedge (\sigma, w \models \varphi_2)$
235	Nono rigo dal basso	φ ⇒ → □φ ⇒
235	Terzo rigo dal basso	modi → mondi
238	Quindicesimo rigo	s0 → s_0
238	Diciassettesimo rigo dal basso	in S) → in s)
238	Dodicesimo rigo dal basso	i≤k'≤k → i≤k'<k
238	Quindicesimo rigo	S0 → S_0
240	Secondo rigo dal basso	Path(S) → Path(s)
241	Primo rigo	S^k → s_k
243	Fig. 14.10	La freccia a destra va da C_i a N_i
244	Secondo punto, secondo rigo	(R)) → (R)))

Per dare la semantica a formule di una logica occorre fare riferimento ad una *struttura di interpretazione*: in questo caso, occorre almeno una funzione di valutazione che dica se una proposizione in questa struttura di interpretazione vale true o false: $V : P \to \{true, false\}$, dove P l'insieme delle proposizioni. Data una formula φ la sua semantica è data da una funzione $\mathcal{S}(\varphi)$, dove $\mathcal{S} : \Phi \to \{true, false\}$, con Φ l' insieme delle formule della logica, definita come:

- $\mathcal{S}(false) = false$
- $\mathcal{S}(p) = V(p)$
- $\mathcal{S}(\sim \varphi) = \sim \mathcal{S}(\varphi)$
- $\mathcal{S}(\varphi_1 \wedge \varphi_2) = \mathcal{S}(\varphi_1) \wedge \mathcal{S}(\varphi_2)$

La naturale estensione della logica proposizionale è il *calcolo dei predicati del primo ordine*. In questa logica si usano gli stessi connettivi logici del calcolo proposizionale, ma le proposizioni sono sostituite da predicati su variabili definite su un dominio di valutazione. La sintassi introduce predicati e funzioni sul dominio dei valori e i quantificatori su tale dominio, rispettivamente il quantificatore universale e il quantificatore esistenziale:

$$\varphi := \sim \varphi \mid \varphi \wedge \varphi \mid true \mid p(exp, .., exp) \mid \forall x : \varphi \mid \exists x : \varphi \tag{14.3}$$

$$exp := x \mid f(exp, .., exp) \tag{14.4}$$

dove x è una variabile su un dominio di valutazione D, p è un predicato $p : D^n \to \{true, false\}$ definito su espressioni sul dominio D, e f è un'operazione su D: $f : D^n \to D$; non andiamo qui oltre nella specifica della sintassi dei predicati e delle funzioni. Vale la dualità tra i due quantificatori (la negazione di una formula universale è una formula esistenziale e viceversa) quindi si potrebbe usare solo uno di essi come operatore primitivo della logica e definire l'altro come operatore derivato:

- $\forall x : \varphi = \sim \exists x : \sim \varphi$

Per definire la semantica della logica dobbiamo introdurre la struttura di interpretazione $\Sigma = (D, V, \chi, \pi)$ dove:

- D è il dominio di interpretazione delle variabili
- V è la funzione di valutazione delle variabili
- χ è la funzione di interpretazione delle funzioni $\chi : F \to (D^n \to D)$
- π è la funzione di interpretazione dei predicati
  $\pi : P \to (D^n \to \{true, false\})$

Definiamo quindi la semantica come una funzione $S : \varphi \to \{true, false\}$, utilizzando anche una funzione ausiliaria di valutazione delle espressioni:
$S' : exp \to D$.

- $S'(x) = V(x)$
- $S'(f(e_1, .., e_n)) = \chi(f)(S'(e_1), , S'(e_n))$
- $S(true) = true$
- $S(\sim \varphi) = \sim S(\varphi)$
- $S(p(e_1, .., e_n)) = \pi(p)(S'(e_1), ., S'(e_n))$
- $S(\exists x : \varphi) = true$ se $\exists y \in D : S(\varphi[x/y]) = true$
- $= false$ altrimenti

dove con $\varphi[x/y]$ si intende la formula φ dove tutte le occorrenze di x siano state rimpiazzate dal valore y. Possiamo ridefinire equivalentemente la semantica attraverso la *relazione di soddisfacibilità* ($\models$):

Σ soddisfa φ ($\Sigma \models \varphi$) quando Σ è un modello per φ, ovvero quando S(φ) è vera, ottenendo così:

- $\Sigma \models true$ sempre
- $\Sigma \models \sim \varphi$ se solo se $\sim \Sigma \models \sim \varphi$
- $\Sigma \models \varphi_1 \wedge \varphi_2$ se solo se $\Sigma \models \varphi_1 \wedge \Sigma \models \varphi_2$
- $\Sigma \models p(e_1, .., e_n)$ se solo se $\Sigma \models \pi(p)(S'(e_1), ., S(e_n)) = true$
- $\Sigma \models \exists x : \varphi$ se solo se $\exists \Sigma' : \Sigma' \models \varphi \wedge \Sigma' = (D, V, \chi, \pi) \wedge V' = V \cup [x \rightarrow y] \wedge y \in D$

Consideriamo ad esempio la formula: $\exists x : p(y, f(y, x))$. Consideriamo una struttura di interpretazione dove D sia l'insieme degli interi positivi non nulli, p sia l'operatore di maggiore $>$, f l'operazione di somma. In questo caso si ha $\exists x : y > y + x = false$ sempre, dunque la struttura presa in considerazione non è un modello per φ.
Se invece consideriamo una diversa struttura di interpretazione per la stessa formula, in cui ho gli stessi operatori, ma $D = \mathbb{Z}$, poiché un valore negativo di x rende la formula quantificata vera, si ha $\exists x : y > y + x = true$, dunque la struttura presa in considerazione è un modello per φ.

Questo esempio mostra come il significato che diamo alla sintassi di una logica può variare a seconda della struttura di interpretazione scelta, cosa che ritroveremo nel seguito. Non faremo più riferimento invece a logiche del primo ordine, perché per gli scopi di questo testo ci basta analizzare alcune logiche definite su base proposizionale, ma che possono essere in principio estese a includere la logica dei predicati del primo ordine. Si noti tra l'altro che la logica del primo ordine permette di esprimere questioni non decidibili, mentre ai fini dell'utilizzo di una logica per fini di verifica formale sarà necessario poter decidere, e con algoritmi efficienti, la soddisfacibilità delle formule su una certa struttura interpretativa.