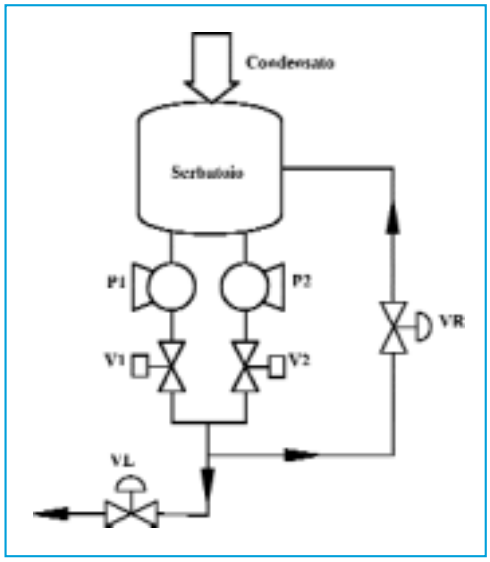
Verifica formale di logiche di controllo di impianti processo



Luca Ferrarini, Francesco Schiavo

Nell'articolo viene esaminata un'applicazione per il controllo di un impianto di energia per illustrarne i vantaggi e proporre possibili soluzioni al problema della verifica formale per sistemi di controllo industriale. L'applicazione riguarda il controllo logico di un sistema di estrazione di acqua condensata per un impianto di energia convenzionale. Partendo dalla teoria della verifica formale e dagli algoritmi di verifica automatica, si passeranno in rassegna le problematiche di modellazione e verifica con riferimento a uno specifico tool (Spin), un checker di modelli logici. Infine, vengono presentati i risultati della verifica completa del sistema.

La complessità delle funzioni di controllo logico, supervisione e protezione realizzate tramite Dcs (Distributed Control System) è, negli ultimi anni, aumentata costantemente per rispondere alle richieste di maggior affidabilità e sicurezza e di prestazioni superiori. Inoltre, l'abbattimento dei costi di progettazione richiede metodologie di sviluppo più sistematiche, che aiutino meglio e in minor tempo a rilevare e rimuovere gli errori a vari livelli, dalla specifica fino all'implementazione finale. Caratteristica particolarmente importante a tal fine è la riutilizzabilità, intesa come strumento per non dover riprogettare funzioni già realizzate e per non dover ripetere le corrispondenti verifiche [2].

A tale aumento di complessità non è tuttavia seguito un miglioramento delle varie fasi della progettazione, e in particolar modo delle fasi di verifica e validazione. Il risultato è un aumento dei tempi richiesti per la progettazione delle logiche e l'installazione del sistema, con ripercussioni anche sui costi di manutenzione per il relativo software di controllo [4, 6]. Infatti, un approccio tipico per la validazione di tali sistemi prevede la realizzazione completa del Dcs stesso, per poi collegare tramite un bus tale sistema a un pannello hardware appositamente realizzato o simulato tramite Pc. Il pannello è equipaggiato con interruttori e pulsanti che un operatore può utilizzare per "simulare" manualmente il processo sotto controllo chiudendo e aprendo gli anelli di controllo o impostando il valore dei segnali sul bus di comunicazione.

Utilizzando tale procedura, le proprietà funzionali e di sicurezza del sistema possono essere testate solo per un insieme limitato di *scenari* che, per quanto scelti in maniera molto accurata, non possono essere esaustivi ed escludere completamente il verificarsi di situazioni non previste, dannose o addirittura pericolose per l'impianto e per gli operatori. Per di più, ciò che viene testato è tipicamente il software di controllo e quindi, nel caso in cui venga rilevato un errore, le cause vanno ricercate sia nell'implementazione sia nelle fasi a monte, senza nessuno strumento per discernerne la natura, rendendo difficoltoso stabilirne la causa. Su tali presupposti si basa la scelta di studiare l'utilizzabilità di tecniche di verifica formale per i sistemi di controllo logico. L'obiettivo è di dimostrare formalmente, cioè matematicamente, che il sistema di controllo progettato abbia le funzionalità richieste e che non siano possibili comportamenti non desiderati, o dannosi. Le procedure di verifica utilizzate in questo lavoro sono basate su tecniche di *model checking*; tali tecniche costituiscono, a oggi, l'approccio

Figura 1 - Sistema di estrazione del condensato

L. Ferrarini, F. Schiavo, Dipartimento di Elettronica e Informazione, Politecnico di Milano, {ferrarin, schiavo}@elet.polimi.it

più promettente per la verifica di sistemi logici [3], anche se le applicazioni per sistemi di controllo sono ancora rare [5, 8].
Il lavoro è organizzato come segue: nel secondo paragrafo viene illustrato il caso di studio industriale (sistema di estrazione del condensato), mentre nel terzo vengono presentati i fondamenti della teoria della verifica formale per sistemi di tipo logico. Il quarto paragrafo è dedicato alla presentazione del modello formale del sistema per uno specifico strumento di analisi (Spin) e il seguente illustra la procedura per la verifica. Vengono infine riassunti i principali risultati raggiunti e presentate le linee di ricerca future.

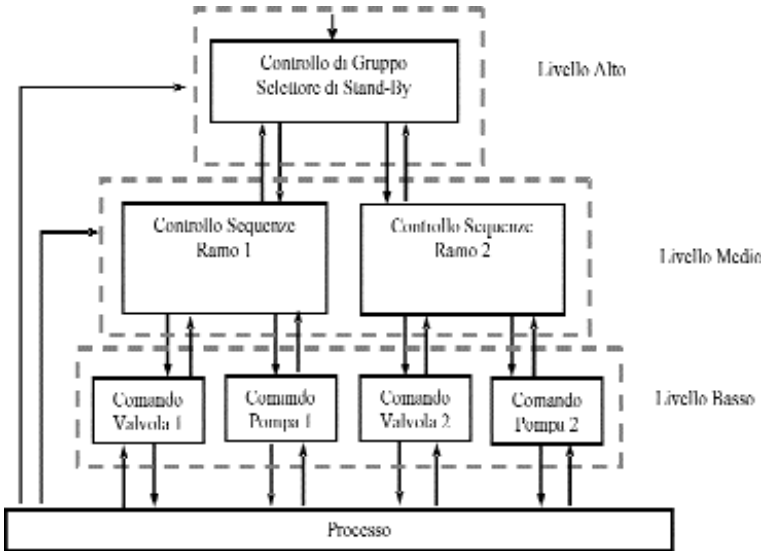


Figura 2 - Schema controllo logico

Il caso di studio: sistema di estrazione del condensato

Il caso industriale utilizzato per testare l'efficacia delle tecniche di verifica formale è una porzione di una centrale termoelettrica. Lo schema semplificato di tale porzione, il sistema di estrazione del condensato, è riportato in Figura 1. I componenti principali del sistema di estrazione sono un serbatoio, due rami di estrazione e una linea di ricircolo. Il condensato in ingresso è raccolto nel serbatoio, dove un sistema di controllo provvede a mantenere il livello costante agendo sulla valvola VL. A valle del serbatoio è presente un sistema di estrazione composto da due rami; in genere solo uno dei due rami è attivo, mentre l'altro è in attesa (o *stand-by*). Ogni ramo è equipaggiato con una valvola di intercettazione manuale e da un filtro meccanico (non rappresentati), oltre che da una pompa (P1, P2) e da una valvola motorizzata (V1, V2). Un secondo sistema di controllo aziona la valvola posta sulla linea di ricircolo (VR), in modo da garantire che il flusso di fluido nelle pompe non scenda sotto il livello minimo durante le fasi di avvio e le operazioni a basso carico dell'impianto.
La porzione di impianto considerata è inoltre dotata di un complesso sistema di automazione e di controllo logico per l'attuazione delle pompe (P1, P2) e delle valvole motorizzate (V1, V2); il sistema è utilizzato prevalentemente durante le fasi di avvio e di spegnimento dell'impianto o durante manovre d'emergenza. L'architettura del sistema di controllo logico è di tipo gerarchico e modulare, organizzata su tre livelli (Figura 2):

- livello alto: controllo dell'avviamento e dello spegnimento dell'intero impianto e controllo della procedura dell'attivazione del ramo in attesa;
- livello medio: controllo delle sequenze di attivazione e di disattivazione dei singoli rami;
- livello basso (o di drive): controllo diretto dei dispositivi fisici (pompe e valvole motorizzate).

In Figura 2 sono riportati solo i principali segnali scambiati; in realtà ogni modulo può scambiare segnali con il campo, con attuatori locali, con console locali o remote e con altre porzioni di logica. Per la verifica del sistema di controllo logico verranno presi in considerazione gli schemi di progetto, costituiti da circa duemila componenti (porte logiche, flip-flop, timer) e comprendenti circa trecentocinquanta segnali (ingressi, uscite e comunicazioni tra moduli) e cento parametri tempo-varianti.

Verifica del sistema di controllo e model checking

Gli schemi di progetto costituiscono un modello del sistema non direttamente analizzabile; per formulare correttamente il problema di verifica è necessario trasformare tale modello del sistema in uno "equivalente" e adatto all'analisi formale [10].
I modelli maggiormente studiati e utilizzati per la verifica formale di sistemi di tipo logico sono basati sulle *strutture di Kripke* [3]. Una struttura di Kripke è un oggetto matematico, simile a un automa, costituito da un insieme di stati (di cui uno indicato come iniziale), da un insieme di transizioni tra stati e da una funzione che etichetta ogni stato con un insieme di proprietà logiche che sono vere

in tale stato. Lo stato iniziale viene determinato in base alle condizioni iniziali del sistema modellizzato. I percorsi nelle strutture di Kripke (cioè sequenze di stati collegati da transizioni) possono rappresentare l'evoluzione temporale di un sistema e le proprietà degli stati "visitati" rappresentano le condizioni del sistema mentre evolve.

L'evoluzione temporale di una struttura di Kripke può essere rappresentata tramite *alberi computazionali*, costruiti "svolgendo" la struttura in un albero di profondità infinita avente come radice lo stato iniziale della struttura stessa. Un albero computazionale rappresenta tutte le possibili evoluzioni del sistema a partire da uno specifico stato iniziale. Gli alberi computazionali costituiscono un formalismo adatto alla formalizzazione di proprietà logiche che si desidera verificare per le corrispondenti strutture di Kripke e quindi per il sistema logico originario [3].

Le proprietà da verificare possono essere espresse tramite una logica temporale del tipo Ctl* (Computation Tree Logic) [9].Le formule appartenenti a Ctl* sono costituite da operatori logici classici (and, or, not) e da operatori speciali: quantificatori di percorso e operatori temporali. Esistono due quantificatori di percorso: *A* (per tutti i percorsi computazionali) e *E* (per almeno un percorso computazionale). Questi quantificatori possono essere utilizzati in riferimento a uno specifico stato per richiedere che tutti i percorsi (o almeno un percorso) che partono da quello specifico stato debbano godere di una proprietà. Esistono cinque operatori temporali, anche se solo due di essi risultano rilevanti per la verifica di sistemi di controllo logico: *F* (eventualmente o in uno stato futuro) e *G* (sempre o globalmente).

Questi operatori possono essere utilizzati per richiedere che una proprietà sia valida in almeno uno stato (o in tutti gli stati) di uno specifico percorso computazionale. I quantificatori di percorso e gli operatori temporali possono essere combinati, insieme ai connettivi logici di base, per costruire differenti tipi di formule. Per comprenderne l'uso, si consideri ad esempio la Figura 3, rappresentante un generico sistema logico dotata di due ingressi (`Apri, Chiudi`) e di due uscite (`Aperto, Chiuso`). Alcune semplici ed interessanti formule temporali per il sistema in esame sono:

- *A G* (not (`Aperto` and `Chiuso`)): con questa formula si richiede che, a partire dallo stato corrente, per ogni percorso computazionale, in ogni stato del percorso è impossibile che `Aperto` e `Chiuso` assumano il valore logico uno contemporaneamente;
- *A* (`Apri` → *F* (`Aperto`)): con questa formula si richiede che, a partire dallo stato corrente, per ogni percorso computazionale, in caso l'ingresso `Apri` assuma il valore uno, prima o poi anche l'uscita `Aperto` assuma il valore uno;
- *E F*(`Chiuso`): con questa formula si richiede che, a partire dallo stato corrente, esista almeno un'esecuzione in cui, prima o poi, l'uscita `Chiuso` valga uno.

È interessante notare che, nonostante i modelli e le formule permettano di descrivere l'evoluzione temporale di un sistema, il tempo non viene gestito in maniera esplicita. Ciò che conta è l'ordine relativo di accadimento degli eventi, non l'istante temporale esatto in cui si sono verificati. Il problema di verificare la validità di una formula temporale per una struttura di Kripke è noto come problema di *model checking* ed è un problema ampiamente studiato dalla comunità scientifica internazionale [3]. Il processo proposto per la verifica formale delle logiche di controllo si articola quindi in tre passi:

- riformulazione del modello del sistema da schemi logici a strutture di Kripke;
- identificazione e specifica delle proprietà da verificare tramite logica temporale;
- verifica della validità delle proprietà "temporali" per le strutture di Kripke tramite algoritmi di model checking.

La terza fase del procedimento può essere effettuata utilizzando moltissimi strumenti e algoritmi già disponibili [3], mentre per le prime due fasi è stato necessario sviluppare una metodologia specifica.

Infatti, la procedura di riformulazione degli schemi logici di controllo tramite strutture di Kripke può essere effettuata manualmente solo per piccole porzioni del sistema, evidenziando quindi la necessità di sviluppare una metodologia automatica o semi-automatica di traduzione. Inoltre, la scelta delle proprietà da verificare non può essere completamente indipendente dal contesto applicativo del sistema di controllo. I prossimi paragrafi sono quindi dedicati alla definizione di una procedura automatica (o automatizzabile) per la rappresentazione degli schemi logici tramite un differente formalismo (un linguaggio di programmazione), basato su strutture di Kripke, e alla identificazione e specifica di un insieme di proprietà, espresse tramite logica temporale, che possano esprimere requisiti di correttezza per le logiche di controllo. Nell'il-

lustrare tali operazioni si fa esplicito riferimento all'ambiente software scelto per la modellizzazione e la verifica: Promela/Spin.

Modellizzazione del sistema tramite Promela/Spin

Lo strumento scelto per la verifica del sistema di controllo logico è Spin [7], uno strumento di model checking molto diffuso, progettato e realizzato interamente nei Bell Labs e liberamente distribuito per scopi educativi e di ricerca [http://spinroot.com].
In Spin, il sistema da verificare è descritto utilizzando un linguaggio chiamato Promela e le proprietà da verificare sono espresse tramite formule appartenenti alla logica Ltl. La logica Ltl è un sottoinsieme della logica Ctl*; la differenza maggiore tra le due consiste nel fatto che ogni formula Ltl deve essere preceduta dal quantificatore di percorso *A*.

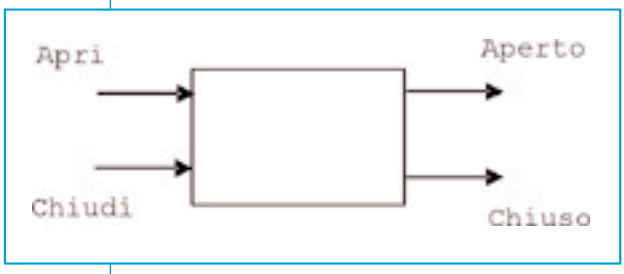


Figura 3 - Sistema logico

Il linguaggio Promela è sintatticamente simile al linguaggio C, e include caratteristiche innovative quali il supporto per l'esecuzione non-deterministica, per la concorrenza tra processi e per la comunicazione e sincronizzazione tra processi.
All'interno di Spin, un programma Promela (per esempio il modello del sistema) viene trasformato automaticamente in una struttura formale equivalente a una struttura di Kripke e adatta alla verifica tramite tecniche di model checking. Il software offre anche la possibilità di simulare il funzionamento del sistema, in modo da poter accertare l'assenza di errori macroscopici prima di effettuare la verifica formale. Il processo di verifica è completamente automatico e può produrre un risultato positivo (cioè il sistema rispetta la proprietà desiderata) o, in caso di errore, generare l'intera sequenza di esecuzione che, a partire dallo stato iniziale, ha portato il sistema in una configurazione in cui la proprietà desiderata non viene rispettata. Tale sequenza può essere utilizzata direttamente dal simulatore in modo da permettere all'utilizzatore di analizzare nel dettaglio l'intera sequenza di eventi che ha portato nella configurazione indesiderata. L'algoritmo di verifica utilizzato all'interno del software si avvale della tecnica di *riduzione parziale dell'ordine* [3]; alla base di tale tecnica vi è la considerazione seguente: nella maggioranza dei casi, ai fini della verifica di una specifica proprietà, l'ordine di esecuzione di due istruzioni concorrenti (che possono rappresentare l'accadimento di due eventi indipendenti) non è significativo.
Risulta quindi possibile effettuare la verifica analizzando solo classi di sequenze di esecuzione: ogni classe rappresenta un intero insieme (di cardinalità potenzialmente infinita) di sequenze indistinguibili ai fini della verifica di una specifica proprietà.
Spin è stato originariamente sviluppato per la verifica di protocolli di comunicazione, tuttavia le potenzialità offerte dai suoi strumenti di modellizzazione e verifica hanno facilitato l'estensione del suo utilizzo anche in contesti molto differenti, con applicazioni anche nel campo della verifica di sistemi di controllo logico e di supervisione [5].
L'uso di tale strumento in quest'ultimo settore richiede, tuttavia, accorgimenti particolari, soprattutto nella fase modellistica: in questo lavoro è stata investigata e sviluppata una metodologia di modellizzazione *ad hoc* per sistemi di controllo logico. Tale metodologia sfrutta le caratteristiche architetturali tipiche dei sistemi in esame: gerarchia e modularità. In particolare, sono stati identificati, sulla base degli schemi logici, tre differenti livelli di aggregazione da rappresentare:
1. componenti logici di base;
2. moduli di controllo logico;
3. comunicazione inter-modulo.

L'approccio modellistico modulare ha permesso di diminuire la complessità del problema (dividendolo in sottoproblemi) e ha portato alla definizione di una corrispondente procedura di verifica modulare/incrementale (verifica dei singoli moduli, dei singoli livelli e dell'intero sistema) e alla definizione di un insieme di proprietà strutturali e funzionali da verificare a ogni livello.

Modellizzazione dei componenti logici di base

I componenti di base sono porte logiche (and, or, not), *flip-flop* e *timer*. Le porte logiche and, or e not e i componenti di tipo flip-flop possono essere rappresentati senza difficoltà

grazie ai costrutti nativi offerti da Promela, mentre la rappresentazione dei componenti di tipo timer richiede alcuni sforzi aggiuntivi. In termini di relazione ingresso/uscita il funzionamento dei componenti di tipo timer è abbastanza semplice: a ogni istante di tempo l'uscita assume il valore uno se l'ingresso ha assunto valore uno (zero) su tutto un intervallo di tempo (di durata definita) precedente; l'uscita assume valore zero in tutti gli altri casi. La rappresentazione di tale relazione è difficoltosa poiché il linguaggio Promela non offre alcun costrutto nativo per rappresentare esplicitamente il tempo. La soluzione adottata si basa su precise assunzioni riguardanti il funzionamento dei componenti, che si ipotizza aderire al seguente schema di tre fasi, ripetuto ciclicamente:
- lettura degli ingressi;
- elaborazione;
- scrittura delle uscite.

Lo schema adottato ha il duplice vantaggio di essere semplice e di approssimare dal punto di vista logico il funzionamento reale dei componenti. I componenti di tipo timer possono quindi essere rappresentati assegnando all'esecuzione di un intero ciclo di tre fasi una durata temporale di riferimento (virtuale): contando il numero di cicli eseguiti è possibile "misurare" il tempo trascorso.

Modellizzazione
dei moduli di controllo logico

Il modello di funzionamento adottato per i moduli di controllo logico è molto simile a quello ciclico e diviso in tre fasi già introdotto per i componenti di base: lettura degli ingressi, elaborazione, scrittura delle uscite.
La composizione dei modelli dei singoli componenti all'interno di un unico programma Promela realizza di fatto le fasi di elaborazione e quella di produzione dei segnali di uscita. La rappresentazione della prima fase è invece più complicata. Infatti, all'inizio di ogni ciclo, il modulo logico dovrebbe leggere i segnali logici (ingressi) provenienti da altre porzioni di logica, dagli operatori o dall'impianto stesso; in particolare, dovrebbe essere prevista la possibilità che, a ogni ciclo di esecuzione, più ingessi indipendenti cambino valore contemporaneamente.
La realizzazione del meccanismo di generazione degli ingressi si è rivelata quindi un aspetto critico, ed è risultata strettamente legata agli algoritmi di verifica utilizzati all'interno di Spin. L'implementazione di tale meccanismo, in seguito adottato anche per la modellizzazione dei livelli e dell'intero sistema, viene discussa ampiamente nel seguito.

Modellizzazione
della comunicazione inter-modulo

La realizzazione di un modello per la comunicazione inter-modulo si basa su di uno schema ben preciso: i modelli dei singoli moduli vengono raggruppati e organizzati secondo lo schema gerarchico indicato negli schemi di progetto; ciò significa che i moduli al livello più alto vengono eseguiti per primi: elaborano gli ingressi e producono le uscite che vengono fornite ai livelli sottostanti. I moduli appartenenti allo stesso livello vengono eseguiti in parallelo e la generazione delle uscite è sincronizzata con la lettura degli ingressi da parte dei moduli del livello inferiore.

Generazione dei segnali
di ingresso in Promela

Per effettuare simulazioni automatiche o verifiche per un singolo modulo, è necessario generare gli opportuni segnali di ingresso. D'altra parte, anche per effettuare simulazioni automatiche o verifiche per l'intero sistema è necessario generare alcuni segnali di ingresso (provenienti dalla console di comando remota, da attuatori locali ecc.).
La strategia alla base di tale operazione (generazione dei segnali di ingresso) è uno dei componenti che maggiormente può influenzare il rapporto tra efficienza e risorse richieste per la verifica formale del sistema. Infatti, la scelta di quali combinazioni o sequenze di ingressi utilizzare non può essere completamente a discrezione dell'utente (non ci sarebbe alcuna garanzia che tutti i casi rilevanti vengano investigati) ma, d'altra parte, la generazione esaustiva di tutte le combinazioni e sequenze di ingressi può rendere non affrontabile la verifica anche per sistemi di dimensione modesta.
Il raggiungimento di entrambi gli obiettivi (efficacia ed efficienza) è stato possibile grazie all'utilizzo in maniera innovativa di alcuni strumenti già presenti in Spin.
Infatti, la metodologia sviluppata sfrutta una particolare sinergia tra alcuni costrutti del linguaggio Promela (per esempio il supporto all'esecuzione non-deterministica) e l'algoritmo di verifica utilizzato all'interno di Spin (tipo riduzione parziale dell'ordine). La descrizione completa e dettagliata delle caratte-

ristiche del linguaggio Promela e dell'algoritmo di riduzione parziale dell'ordine non è tra gli scopi di questo lavoro, tuttavia viene qui fornita una semplice introduzione agli argomenti sopraddetti in modo da poter presentare la metodologia ideata.
All'interno di porzioni di codice Promela è possibile eseguire una singola istruzione scelta in maniera casuale all'interno di un insieme di istruzioni tutte potenzialmente eseguibili ma mutuamente esclusive. Utilizzando tale caratteristica all'inizio di ogni ciclo di esecuzione, è possibile generare potenzialmente tutte le combinazioni e sequenze di ingresso per un modulo logico semplicemente inserendo, per ogni segnale di ingresso, l'assegnamento ai due possibili valori *(vero*, *falso)* in un insieme di istruzioni potenzialmente eseguibili ma mutuamente esclusive.
Si consideri, ad esempio, la seguente porzione di pseudo-codice che genera i segnali di ingresso `IN1` e `IN2`:

```
inizioCiclo
{
SceltaDiUnaIstruzione
      {
      IN1:=VERO;
      IN1:=FALSO;

      IN2:= VERO;
      IN2:=FALSO;
}
…
}
fineCiclo
```

All'inizio di ogni ciclo, le quattro istruzioni di assegnamento per i due ingressi sono tutte eseguibili, ma solamente una verrà effettivamente eseguita: in questo modo a ogni ciclo uno dei segnali di ingresso (scelto in maniera casuale) potrebbe cambiare valore.
Per generare potenzialmente tutte le combinazioni e sequenze di ingresso è necessario che ciascun segnale di ingresso, a ogni ciclo, possa cambiare valore indipendentemente dagli altri. Per fare ciò basterebbe creare un insieme di due istruzioni eseguibili e mutuamente esclusive per ogni segnale di ingresso; tale soluzione, tuttavia, porterebbe a un aggravio significativo del carico computazionale per la verifica, dato che ogni insieme aggiunto può potenzialmente raddoppiare il numero di scenari da analizzare.
Una delle soluzioni più diffuse è quella di adottare un'ipotesi semplificativa, supponendo che, a ogni ciclo, solo uno dei segnali di ingresso possa cambiare [8].
La soluzione adottata nell'ambito di questo lavoro è leggermente differente: i segnali di ingresso vengono divisi in opportuni gruppi e, a ogni ciclo di esecuzione, solo un segnale per gruppo può, potenzialmente, cambiare valore. La divisione dei segnali in gruppi è basata sulla provenienza dei segnali stessi; infatti, è molto improbabile che due segnali provenienti da un'interfaccia operatore cambino esattamente nello stesso istante, mentre ciò non è più vero se si considera, ad esempio, un segnale proveniente da un operatore e un segnale proveniente dall'impianto.
La scelta del numero di gruppi in cui dividere gli ingressi è critica, poiché rappresenta il *trade-off* tra generalità del modello da verificare e carico computazionale.
Tramite l'utilizzo di tale tecnica di generazione degli ingressi, non è più necessario "immaginare" o scegliere gli scenari da analizzare per la verifica, dato che l'algoritmo di riduzione parziale dell'ordine utilizzato in Spin riconoscerà automaticamente tutte le classi di esecuzioni rilevanti per la verifica di ogni singola proprietà.

Modellizzazione del processo controllato

Il sistema di controllo logico è strettamente collegato al processo controllato, in modo che il loro funzionamento dipenda dalla loro mutua influenza; è quindi necessario investigare anche quali possano essere i modelli del processo controllato adatti alla verifica formale del sistema di controllo logico secondo la metodologia e con gli strumenti sopra indicati. Le classi di modelli utilizzabili si dividono in due categorie principali: modelli in anello aperto (OL), dove solo il controllo logico viene rappresentato e modelli in anello chiuso (CL), dove vengono modellizzati il controllo logico, il processo controllato e la loro mutua influenza.
La prima soluzione (OL) ha l'indubbio vantaggio di essere più semplice (non è necessario realizzare alcun modello per l'impianto) ed è maggiormente indicata per verifica di proprietà strutturali del sistema di controllo logico, proprietà che il sistema di controllo deve possedere indipendentemente dal processo controllato. Esempi di queste proprietà sono l'attivazione delle uscite (tutti i segnali logici devono avere la possibilità di essere attivati) e la coerenza dei comandi (non deve essere possibile generare comandi logicamente incoerenti, come l'attivazione e disattivazione contemporanea di un attuatore).

La seconda soluzione (CL) ha il vantaggio di essere maggiormente descrittiva, permettendo di verificare anche proprietà la cui validità dipende dall'interazione tra le logiche di controllo e l'impianto.
Un esempio di tale proprietà è l'attivazione di un segnale da parte dell'impianto atto a informare il controllo logico della variazione di una variabile fisica in risposta a una precedente azione del controllo stesso (si pensi al segnale *"livello_serbatoio_basso"* attivato dopo che una pompa di estrazione è stata in funzione per qualche tempo). L'aspetto negativo di questa seconda soluzione è costituito dalla necessità stessa di realizzare un modello per il processo controllato: molto spesso gli strumenti di model checking non offrono la possibilità di realizzare modelli per le dinamiche continue che stanno alla base del processo controllato. Una soluzione di tipo CL meno complicata è costituita dalla realizzazione di semplici modelli logici solo per le parti di impianto che scambiano effettivamente informazioni con il sistema di controllo logico, trascurando la modellizzazione delle dinamiche fisiche del processo.
All'interno di Spin, l'unica soluzione CL adottabile è quest'ultima, a causa delle limitazioni del linguaggio Promela e del sistema di verifica, realmente efficiente solo quando il modello è completamente logico. Nell'ambito del presente lavoro le soluzioni adottate sono sempre di tipo OL e, ove richiesto dalla particolare natura delle proprietà da verificare, di tipo CL con semplici modelli logici per le valvole e le pompe controllate.

Verifica del modello

In questo paragrafo viene affrontato il problema della verifica formale per il modello realizzato; viene dapprima esaminato il problema dell'identificazione di opportuni insiemi di proprietà che il sistema deve possedere e in seguito, tramite l'applicazione al caso di studio, viene illustrata la procedura di verifica modulare e incrementata la proposta.

Formule temporali

L'utilizzo di formule temporali per rappresentare le proprietà desiderate permette, in linea di principio, di condurre sul sistema tipi di analisi anche molto diversi tra loro [11]; è quindi opportuno identificare le categorie di proprietà più significative per i sistemi di controllo logico, specificando sia proprietà di tipo strutturale, indipendenti cioè dal tipo di funzione logica svolta, sia proprietà di tipo funzionale, dipendenti cioè dalla specifica funzione logica svolta.
La prima categoria di proprietà strutturali da verificare riguarda l'attivazione delle uscite di ogni porzione logica: gli schemi di controllo logico non devono garantire che ogni segnale logico di uscita non sia bloccato, possa cioè assumere sia il valore logico uno sia zero. Per garantire la validità di tale proprietà, è possibile verificare che le seguenti formule temporali siano false (`OUT` è un generico segnale logico di uscita):
- $A\,G$ (`OUT`);
- $A\,G$ (not `OUT`).

Affinché la prima formula sia valida, è necessario che il segnale `OUT` assuma il valore uno per ogni percorso computazionale. Dualmente, affinché la seconda formula sia valida, è necessario che il segnale `OUT` assuma il valore zero per ogni percorso computazionale. La seconda categoria di proprietà strutturali coinvolge la verifica per porzioni di logica che gestiscono segnali di comandi del tipo on/off, come, ad esempio, i comandi dati agli attuatori o i comandi per le sequenze di avviamento e spegnimento. Tramite la verifica è possibile controllare che i comandi generati non siano incoerenti: non deve essere possibile che entrambi i comandi assumano contemporaneamente il valore uno. La proprietà viene verificata dimostrando la validità della seguente formula temporale:
- $A\,G$ not (`ON` and `OFF`).

La verifica di proprietà di tipo strutturale deve essere seguita dalla verifica di proprietà di tipo funzionale, proprietà quindi che dipendono dalla specifica funzione logica svolta dagli schemi sotto esame. Ad esempio, è desiderabile poter verificare che, qualora venisse dato un comando di chiusura di emergenza per una valvola, tale comando venga effettivamente eseguito. Più in generale è necessario poter verificare che, in situazioni specifiche, il sistema si porti in una specifica configurazione. La formula temporale per esprimere tale richiesta è la seguente (formula di risposta):
- $A\,G\,(f \Rightarrow (F\,g))$.

In tale formula f rappresenta le condizioni logiche che dovrebbero far si che il sistema "reagisca" portandosi nella configurazione espressa dalle condizioni logiche g. Affinché la formula temporale sia vera è necessario che, per ogni percorso computazionale, qua-

lora esista uno stato in cui sono valide le condizioni *f*, allora esiste anche uno stato in cui sono valide le condizioni *g*. In realtà estensioni di tale "formula di risposta" permettono di formulare specifiche riguardanti conteggi di eventi, sequenze di eventi, serie di sequenze e paralleli di sequenze.

Procedura di verifica

La procedura di verifica segue il paradigma gerarchico/modulare utilizzato anche nella modellizzazione e derivato dagli schemi logici stessi. La procedura è quindi divisa in tre fasi sequenziali:
- verifica dei singoli moduli;
- verifica dei livelli;
- verifica dell'intero sistema.

La divisione della procedura di verifica in tre fasi può apparire inutilmente complicata, dato che l'obiettivo finale è quello di effettuare la verifica per l'intero controllo logico. Al contrario, la scelta di operare tramite fasi di complessità crescente aiuta ad affrontare le difficoltà dell'intero problema; infatti le prime due fasi sono "propedeutiche" per la terza, identificando errori e inconsistenze per porzioni di logica relativamente piccole e permettendo, durante la terza fase, di focalizzare la verifica solo sulle interazioni tra tali porzioni logiche. Inoltre, l'approccio modulare ha permesso di diminuire il carico computazionale per le fasi di verifica dell'intero sistema, dato che, per numerose proprietà, la validità delle corrispondenti formule temporali in una fase implica la validità anche nelle fasi successive.

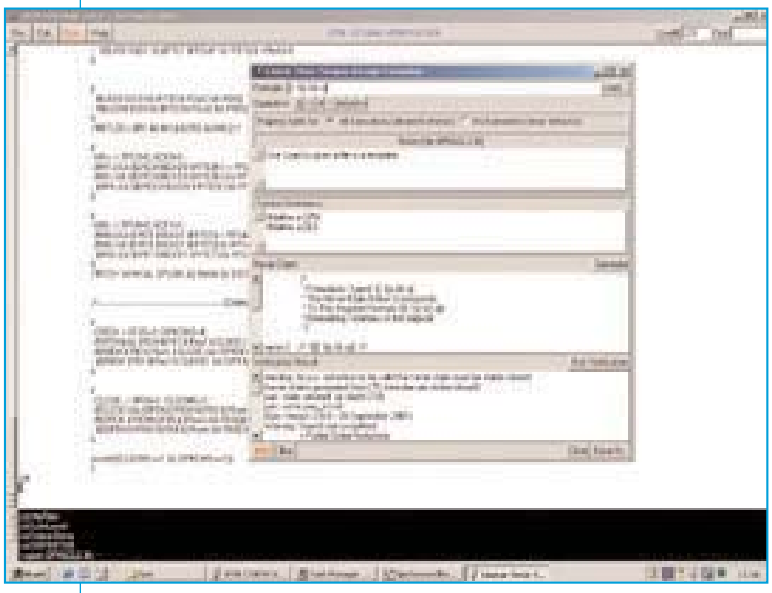


Figura 4 - Verifica con Spin

Risultati

Seguendo la procedura di modellizzazione sopra introdotta, è stato costruito, tramite Promela/Spin, un modello completo per l'intera logica di controllo del caso di studio presentato (Figura 4).
È stato inoltre identificato un vasto insieme di proprietà da verificare, sia di tipo strutturale sia di tipo funzionale. La verifica formale ha condotto ai risultati seguenti.
- Verifica dei singoli moduli logici
  - elevato numero di proprietà verificate;
  - asso sforzo computazionale (<15 sec/prop).
- Verifica dei singoli livelli
  - elevato numero di proprietà verificate;
  - crescita significativa dello sforzo computazionale (≈20 min/prop).
- Verifica dell'intero sistema
  - carico computazionale eccessivo.

I risultati delle prime due fasi di verifica sono molto incoraggianti: numerosi errori "nascosti" e inconsistenze sono stati rilevati e corretti; al contrario, il carico computazionale eccessivo non ha permesso di portare a termine la terza fase. Nonostante i problemi di complessità computazionale, dovuti anche all'oggettiva complessità del sistema analizzato, l'analisi formale effettuata ha permesso di migliorare notevolmente gli schemi logici di controllo, rimuovendo numerosi possibili malfunzionamenti le cui cause sarebbero state estremamente difficili da identificare per il sistema installato.

Conclusioni e sviluppi futuri

In questo lavoro si è studiata l'applicazione di tecniche di verifica formale, e in particolare della tecnica di *model checking*, a sistemi di controllo logico. Le tecniche studiate sono state applicate a un caso di studio tratto dalla realtà industriale. La metodologia di realizzazione di un modello orientato alla verifica per sistemi di controllo logico è stata ampiamente discussa, sia da un punto di vista teorico, sia in riferimento a uno specifico strumento (Spin); i risultati ottenuti possono comunque essere facilmente estesi per altri strumenti o applicazioni.

L'identificazione di categorie di proprietà, sia strutturali sia funzionali, di cui verificare la validità ha inoltre costituito parte significativa del lavoro. I risultati ottenuti dall'applicazione della metodologia proposta a un caso di studio sono stati presentati e discussi. Tali risultati si sono rivelati incoraggianti, nonostante alcune limitazioni dovute alla complessità computazionale per la verifica dell'intero sistema.
Le direzioni di ricerca future sono orientate verso una più approfondita analisi e comprensione dei limiti computazionali delle tecniche proposte e a un'estensione del paradigma modellistico e di verifica in termini di gestione esplicita di variabili temporali.

Bibliografia

[1] Bemporad A., M. Morari, *Control of systems integrating logic, dynamics, and constraints,* Automatica 35, n. 3, 407-427, 1999.

[2] Bowen J.P., M.G. Hinchey, *The use of industrial-strength formal methods*, In Proceedings of the Computer Software and Applications Conference (Aug 11-15), Ieee, 1997.

[3] Clarke E.M., O. Grumberg, D.A. Peled, *Model Checking*. The Mit press, Cambridge, Massachusetts, 2000.

[4] De Smet O., J.J. Lesage, J.M. Roussel, *Formal verification of industrial control systems,* In Proceedings of the 10th Ifac Symposium on Information Control Problems in Manufacturing (Vienna University of Technology, Vienna, Austria), vol. 1, September 2000, pp. 125-132, Ifac, 2000.

[5] Havelund K., M.R. Lowry, J. Penix, *Formal analysis of a space-craft controller using Spin*, Software Engineering 27, n. 8, 1072-1095, 2001.

[6] Heitmeyer C.L., 1998, *On the need for practical formal methods,* Formal Techniques in Real-Time and Real-Time Fault-Tolerant Systems, Proc., 5th Intern. Symposium (Ftrtft '98), Lyngby, Denmark, September 14-18, pp. 18-26, 1998.

[7] Holzmann G.J, *The model checker Spin*, Software Engineering 23, n. 5, 279–295, 1997.

[8] Park,T., 1997, *Formal verification and dynamic validation of logic-based control systems,* Ph.D. thesis, Massachusetts Institute of Technology, 1997.

[9] Pnueli A., 1977, *The temporal logic of programs,* In Proceedings of the 18th Annual Symposium on Foundations of Computer Science, Ieee Computer Society Press, pp. 46–57, 1977.

[10] F. Schiavo, 2002, *Tecniche di verifica formale per le logiche di controllo di impianti di processo,* Tesi di Laurea, Politecnico di Milano, a.a. 2001/2002.

[11] A.K. Zaidi, A.H. Levis, *Validation and verification of decision making rules,* Automatica 33, n. 2, 155-169, 1997.