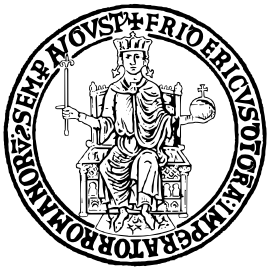
Università degli Studi di Napoli Federico II



Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione

Corso di Laurea Magistrale in Informatica

Intuitionistic Computation Tree Logic

Relatori
Ch.mo Prof. Aniello Murano

Correlatore
Dott. Davide Catta

Controrelatore
Ch.mo Prof. Andrea Calì

Candidato
Andrea Capone
N97000359

Anno Accademico 2023–2024

Università degli Studi di Napoli Federico II

Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione

Corso di Laurea Magistrale in Informatica

Intuitionistic Computation Tree Logic

Relatori
Ch.mo Prof. Aniello Murano

Correlatore
Dott. Davide Catta

Controrelatore
Ch.mo Prof. Andrea Calì

Candidato
Andrea Capone
N97000359

Anno Accademico 2023–2024

Abstract - Inglese

My thesis presents an intuitionistic version of Computation Tree Logic (CTL), known as Intuitionistic Computation Tree Logic (ICTL). I focused my efforts on the development and understanding of ICTL, which merges intuitionistic logic with CTL. A distinctive feature of ICTL is its birelational models and syntax that clearly distinguishes between existential and universal quantifications.
In my research, I outlined the syntax and semantics of ICTL, examined the properties of birelational models, and verified the satisfiability of ICTL formulas. I explored the differences between CTL and ICTL, particularly the validity of fixed-point equivalences, introducing a new version of ICTL with an additional condition on the models to overcome limitations of the original proposal. I developed an efficient model checking algorithm for this new version of ICTL, highlighting its computational complexity and practical applications in constructively verifying system properties.
In my thesis, I emphasized the importance of ICTL in verifying properties across various contexts, including open systems, and highlighted its potential for addressing more complex verification tasks compared to classical CTL. I discussed ICTL's application in different scenarios such as medical monitoring, database management, and agricultural land monitoring, emphasizing its ability to ensure persistent knowledge and information traceability over time.
Finally, I presented an extension of VITAMIN, a prototype designed for formal verification of Multi-Agent Systems (MAS), highlighting its modularity, versatility, and ease of extension to accommodate various logics and models in property verification. I provided benchmarks to demonstrate the efficiency of the ICTL model checking process in VITAMIN. I hope that my thesis can contribute to the existing literature on CTL and stimulate further research in this field.

Abstract - Italiano

La mia tesi presenta una versione intuizionista della logica temporale CTL, conosciuta come Intuitionistic Computation Tree Logic (ICTL). Ho concentrato i miei sforzi nello sviluppo e nella comprensione di ICTL, che unisce la logica intuizionista e la Computation Tree Logic (CTL). Una caratteristica distintiva dell'ICTL sono i suoi modelli birelazionali e una sintassi che separa chiaramente le quantificazioni esistenziali e universali.
Nella mia ricerca, ho delineato la sintassi e la semantica dell'ICTL, esaminato le proprietà dei modelli birelazionali e verificato la soddisfacibilità delle formule dell'ICTL. Ho esplorato le differenze tra CTL e ICTL, in particolare la validità delle equivalenze dei punti fissi, introducendo una nuova versione di ICTL con una condizione aggiuntiva sui modelli per superare le limitazioni della proposta originale. Ho sviluppato un algoritmo efficiente di model checking per questa nuova versione di ICTL, evidenziando la sua complessità computazionale e le sue applicazioni pratiche nella verifica costruttiva delle proprietà dei sistemi.
Nella mia tesi, ho enfatizzato l'importanza di ICTL nella verifica delle proprietà in vari contesti, inclusi i sistemi aperti, e ho evidenziato il suo potenziale per affrontare compiti di verifica più complessi rispetto alla classica CTL. Ho discusso l'applicazione dell'ICTL in diversi scenari, come il monitoraggio medico, la gestione dei database e il monitoraggio dei terreni agricoli, sottolineando la capacità di ICTL di garantire conoscenze persistenti e tracciabilità delle informazioni nel tempo.
Infine, ho presentato un'estensione di VITAMIN, un prototipo progettato per la verifica formale dei Sistemi Multi-Agente (MAS), evidenziandone la modularità, la versatilità e la facilità di estensione per ospitare diverse logiche e modelli nella verifica delle proprietà. Ho fornito benchmark per dimostrare l'efficienza del processo di model checking ICTL in VITAMIN. Spero che la mia tesi possa contribuire alla letteratura esistente sulla CTL e stimolare ulteriori ricerche in questo campo.

Introduzione

I **metodi formali** [1] sono tecniche matematiche utilizzate per la specifica, lo sviluppo e la verifica di sistemi software e hardware. Questi metodi permettono di descrivere e analizzare i sistemi in modo rigoroso e preciso, riducendo il rischio di errori e aumentando la fiducia nella loro correttezza. Basandosi su un formalismo matematico per rappresentare gli stati e le transizioni di un sistema, i metodi formali includono l'uso di linguaggi di specifica formale, logiche matematiche e modelli di calcolo. Offrono strumenti per la verifica formale, il processo di dimostrazione matematica che un sistema soddisfa una specifica data.
I metodi formali sono fondamentali in campi dove la sicurezza e l'affidabilità sono cruciali, come nei sistemi critici di controllo aerospaziale, nelle reti di comunicazione, nei sistemi finanziari, nei dispositivi medici e nei sistemi multi-agente [2], [3]. Questi metodi permettono di **individuare** e **correggere** errori nelle fasi iniziali del processo di sviluppo, fornire una descrizione **precisa** e **non ambigua** dei requisiti e delle specifiche del sistema, e dimostrare la **correttezza** del sistema rispetto alle specifiche date.
L'importanza dei metodi formali risiede nella loro capacità di **migliorare** significativamente la **qualità** e **l'affidabilità** dei sistemi complessi. L'uso di tecniche matematiche rigorose permette di gestire la complessità dei sistemi moderni e di garantire che essi funzionino correttamente anche in condizioni estreme. Tuttavia, uno **svantaggio** principale di questi metodi è l'elevata quantità di risorse computazionali richieste, il che può renderne difficile l'applicazione su larga scala. Inoltre, modellare sistemi complessi con metodi formali può risultare estremamente complicato e dispendioso in termini di tempo, a causa della necessità di descrivere ogni dettaglio del sistema in modo rigoroso e preciso. Pertanto, è importante individuare i casi in cui vale davvero la pena applicare le tecniche di verifica e valutare se il problema può essere affrontato utilizzando tecniche di verifica più leggere possibile. Questo approccio permette di ottimizzare le risorse e ridurre i costi computazionali, riservando la verifica formale per situazioni dove la precisione e la completezza dei sistemi sono imprescindibili. In altre circostanze, tecniche di verifica meno onerose possono essere sufficienti per garantire un livello adeguato di affidabilità e correttezza del sistema, senza incorrere nell'elevato costo computazionale associato a questi metodi.
Una delle metodologie formali più rilevanti e leggere è il **model checking** [4], [5], una tecnica automatica di verifica formale utilizzata per controllare se un modello di sistema soddisfa una specifica data. Questa tecnica permette di esplorare tutti i possibili stati

e transizioni di un sistema in modo sistematico, verificando la validità delle proprietà desiderate senza dover eseguire manualmente l'intero processo. Il model checking consiste nel costruire un modello formale del sistema e una specifica formale delle proprietà ,espresse attraverso dei formalismi logici, che il sistema deve soddisfare. Il model checker esplora tutte le possibili configurazioni del modello per determinare se le specifiche sono soddisfatte e, in caso contrario, fornisce controesempi che dimostrano dove e come il sistema viola le proprietà specificate.
Le **logiche temporali** [6] sono formalismi logici utilizzati per descrivere e analizzare i comportamenti temporali dei sistemi. Esse permettono di esprimere proprietà che descrivono come i sistemi si evolvono nel tempo, facilitando la verifica e la modellazione della dinamica dei sistemi. Le logiche temporali sono particolarmente utili nei contesti in cui il comportamento del sistema dipende da sequenze di eventi o stati che si verificano nel tempo.
Una logica temporale di rilievo è la **CTL (Computation Tree Logic)**, che permette di esprimere proprietà sui comportamenti dei sistemi lungo tutti i possibili cammini computazionali. A differenza di LTL (Linear Time Temporal Logic), che considera singoli cammini lineari, CTL considera la struttura ad albero delle possibili esecuzioni di un sistema. Poiché il model checking di proprietà CTL è un problema risolvibile in tempo polinomiale, questa tecnica risulta essere una delle più utilizzate nella verifica formale.
Le proprietà del **punto fisso** [7], [8], sono essenziali per calcolare la verifica delle formule CTL. Infatti, alcuni operatori temporali in CTL possono essere definiti in termini di punti fissi. L'algoritmo di model checking CTL sfrutta queste proprietà dei punti fissi per determinare se una formula è soddisfatta in un modello. Questo approccio permette di verificare automaticamente proprietà complesse su sistemi dinamici, inoltre, sfruttando le proprietà dei punti fissi, il model checking risulta avere una complessità di tempo **lineare** sul modello e, dato che la soddisfacibilità di una formula CTL dipenderà dalla soddisfacibilità delle sue sottoformule, è possibile dimostrare che il model checking richiede tempo lineare anche sulla lunghezza della formula in input per un totale di $O(|\mathcal{M}| * |\phi|)$.
La verifica formale potrebbe essere utile anche in contesti in cui è importante **tracciare l'informazione** perché garantisce che i processi e le transazioni siano accurati e coerenti. La verifica formale utilizza metodi matematici per dimostrare che un sistema soddisfa determinate proprietà specifiche, riducendo la possibilità di errori e ambiguità. In un ambiente in cui la tranciabilità delle informazioni è cruciale, come nel settore finanziario, sanitario o logistico, la verifica formale può assicurare che ogni passaggio del processo sia verificabile e affidabile. Questo aumenta la trasparenza e la fiducia nelle informazioni tracciate, migliorando la gestione del rischio e la conformità alle normative.
L'intuizionismo [9], [10] è una corrente della logica matematica che sostiene che le verità matematiche non siano scoperte, ma piuttosto costruite dalla mente umana [11]. Secondo l'intuizionismo, una proposizione è vera solo se esiste una prova costruttiva della sua verità. Questo si contrappone al realismo platonico, che vede le entità matematiche come esistenti indipendentemente dal nostro pensiero.
L'intuizionismo è collegato al concetto di informazione [12] e conoscenza di un agente in quanto enfatizza il ruolo delle prove costruttive e del processo di conoscenza. Per un

intuizionista, sapere qualcosa significa avere una prova esplicita e costruttiva di quel qualcosa. Questo concetto è in linea con l'idea che l'informazione e la conoscenza siano strettamente legate ai metodi e ai processi attraverso i quali vengono ottenute. In contesti come l'informatica e l'intelligenza artificiale, dove la gestione dell'informazione e la tranciabilità dei processi sono fondamentali, l'intuizionismo fornisce un quadro teorico per garantire che le informazioni siano non solo disponibili, ma anche costruttivamente verificabili e affidabili.
A tal fine, è stato sviluppato un formalismo logico che unisce l'intuizionismo al concetto di "information gain" insieme alla logica CTL, creando così una versione intuizionista di CTL nota come ICTL [13]. Questa logica consente di descrivere due aspetti temporali dei modelli: uno riguardante la dinamica del sistema in base alle sue computazioni, e l'altro focalizzato sulla conoscenza di un agente esterno che osserva il sistema.
In questo contesto, ICTL permette di rappresentare in modo preciso e formale sia l'evoluzione dei modelli nel tempo, secondo una logica temporale ben definita, sia la percezione e l'acquisizione di conoscenza da parte di un agente esterno. Questo approccio non solo enfatizza l'importanza della costruttività nella logica, in linea con i principi intuizionisti, ma integra anche la **gestione** dell'informazione e la **tracciabilità** dei processi, fondamentali per l'analisi e lo sviluppo di sistemi complessi e affidabili, come quelli impiegati nei settori tecnologici avanzati e nella sicurezza informatica.
In letteratura è già presente un formalismo ICTL, tuttavia, si è riscontrato che questo formalismo non valida le proprietà del punto fisso per alcuni operatori temporali. Questo significa che in certi casi le caratteristiche essenziali dei punti fissi, cruciali per l'algoritmo di model checking, non sono adeguatamente trattate o garantite.
Nel mio lavoro di tesi, mi sono occupato di sviluppare una versione avanzata di ICTL che introduce una condizione sui modelli aggiuntiva rispetto a quelle già presenti, per assicurare che le proprietà del punto fisso siano sempre valide. Questo approccio è fondamentale per garantire che la logica rimanga robusta e coerente anche nei casi più complessi e per fornire una base solida per l'analisi e la verifica dei sistemi basati su ICTL.
Questo miglioramento non solo aiuta a superare le lacune presenti nel formalismo precedente, ma rafforza anche il ruolo di ICTL come uno strumento affidabile ed efficace per la modellazione e la verifica formale dei sistemi, inclusa l'analisi delle loro proprietà dinamiche. Inoltre, questo sviluppo contribuisce significativamente alla capacità di modellare l'informazione che si acquisisce durante l'osservazione del modello.
Il principale vantaggio di questa logica risiede nel fatto che, nonostante consenta di gestire un maggiore livello di informazioni rispetto a CTL, l'algoritmo di model checking associato a questa nuova logica conserva la **stessa** complessità computazionale. Ciò significa che rimane trattabile dal punto di vista computazionale.
Questa caratteristica è cruciale perché garantisce che, nonostante la capacità espansa di rappresentare e analizzare informazioni più dettagliate e complesse, l'analisi formale dei modelli rimanga efficiente e gestibile. Di conseguenza, ICTL rappresenta un avanzamento significativo nel campo della verifica formale dei sistemi, consentendo di mantenere elevati standard di precisione e affidabilità senza compromettere la scalabilità computazionale, essenziale per l'applicazione pratica in settori quali la sicurezza informatica, la proget-

tazione di sistemi critici e i sistemi multi agente.
Infine, per validare questo lavoro di ricerca, ho sviluppato e implementato un software per il model checking di ICTL, in particolare un'estensione del tool **VITAMIN** [14], un tool di model checking per sistemi multi-agente. I risultati ottenuti e discussi nell'elaborato sono stati estremamente positivi.
L'implementazione software ha dimostrato l'efficacia e la praticità del formalismo ICTL nella pratica, confermando la capacità di gestire in modo efficiente e accurato la verifica formale dei sistemi e delle loro dinamiche temporali. Questo passaggio cruciale non solo ha consolidato la teoria sviluppata, ma ha anche fornito una piattaforma concreta per l'applicazione della logica ICTL in contesti reali.
L'implementazione software rappresenta quindi un passo significativo verso l'adozione e l'uso diffuso di ICTL come strumento avanzato per la modellazione e l'analisi formale di sistemi complessi, contribuendo così alla crescita e all'innovazione nel campo della logica temporale e della verifica formale.

Sommario

La tesi si articola attraverso vari capitoli dedicati a esplorare e approfondire diversi aspetti della logica intuizionista e delle sue applicazioni nel contesto della logica modale e del model checking temporale. Il primo capitolo introduce la logica intuizionista, delineandone le motivazioni storiche come alternativa alla logica classica. Si esplorano i concetti fondamentali e le proprietà principali della logica intuizionista, supportati da dimostrazioni che ne sottolineano l'applicazione pratica e la validità teorica. Si esplora poi la logica intuizionista modale. Qui si introduce il concetto di operatori modali. Il secondo capitolo è dedicato alla logica Computation Tree Logic (CTL), una forma di logica temporale cruciale per la verifica delle proprietà nei sistemi software nel tempo. Si esplora la sintassi e la semantica della CTL, inclusa l'importanza del model checking per verificare proprietà temporali nei modelli computazionali. Il terzo capitolo introduce ICTL, una variante di CTL che incorpora principi della logica intuizionista. Si discute la validità di formule specifiche in ICTL, le differenze rispetto alla CTL e le sfide legate alle formule temporali universali. Successivamente viene introdotta ICTL, che aggiunge una condizione al modello per superare alcune limitazioni dell'ICTL originale. Questo permette la verifica locale delle formule universali mantenendo la validità delle equivalenze del punto fisso che nella logica inizialmente introdotta non risultavano essere formule valide. Si delinea poi un approccio efficiente per il model checking di ICTL, che coinvolge l'analisi della verità delle formule su ogni stato e il calcolo dei punti fissi, sottolineando l'importanza di tecniche pratiche nella verifica formale. Nel quarto capitolo, si esplora come la logica intuizionista possa essere estesa per affrontare concetti di informazione in termini di prova teorica o algoritmo, che sono poi tradotti in specifiche riguardanti il tracciamento dell'informazione durante il processo di computazione del sistema. Questo approfondimento permette di analizzare come la logica intuizionista, con il suo focus sulla costruzione di prove e sulla derivazione di informazioni da presupposti intuitivi anziché da verità assolute, possa essere applicata in contesti in cui è cruciale monitorare e tracciare l'evoluzione dell'informazione nel tempo. Successivamente il capitolo esplora le applicazioni pratiche di ICTL in vari contesti, come il monitoraggio medico, le operazioni di scrittura su database e il monitoraggio di terreni agricoli, evidenziando come la teoria possa essere applicata in settori concreti. Infine, il si discute l'implementazione e il benchmarking di ICTL in VITAMIN, uno strumento di model checking progettato per verificare Sistemi Multi-Agente (MAS) utilizzando un modello basato su logica temporale. In sintesi, la tesi non solo esplora le fondamenta teoriche della logica intuizionista e

modale, ma anche come queste teorie possano essere applicate attraverso tecniche di verifica formale, evidenziando il loro impatto in diversi settori applicativi.

Contents

–1–
Intuitionistic Propositional Logic

CONTENTS: **1.1 Intuizionismo. 1.2 Sintassi IPL. 1.3 Semantica IPL. 1.4 Logica intuizionista modale.**

In questo capitolo, sono riportati dei cenni alla logica intuizionista, delineando le ragioni storiche che hanno portato alla sua nascita come un'alternativa all'approccio della logica classica, mettendo in luce le principali differenze tra i due approcci. Verranno inoltre introdotti i concetti fondamentali che ne costituiscono le basi, consentendo così una comprensione approfondita della sua natura e delle sue implicazioni. Saranno esposte le proprietà principali della logica intuizionista, accompagnate da **dimostrazioni rigorose svolte durante** la stesura dell'elaborato che ne illustrano l'applicazione e la validità. Infine, verrà esplorata l'evoluzione storica della logica intuizionista modale e le motivazioni che hanno guidato lo sviluppo dei suoi modelli, offrendo così una panoramica generale del suo funzionamento.

1.1 Intuizionismo

L'intuizionismo [15] è una scuola di pensiero matematico sviluppata nei primi del '900 dal matematico olandese **L.E.J. Brouwer** [10] la quale respinge l'idea tradizionale che il valore di verità di una proposizione matematica sia indipendente dalla nostra capacità di conoscerla o di verificarla. In altre parole, gli intuizionisti sostengono che i valori di verità di un'affermazione matematica siano le sue condizioni di dimostrabilità. Questa prospettiva mette in discussione alcuni dei principi fondamentali della logica classica e della matematica convenzionale [11]:

- principio del *terzo escluso* **(LEM)** $A \vee \neg A$;
- eliminazione della *doppia negazione* **(DNE)**; $\neg\neg A \to A$;

tuttavia, ammette:

- il principio della *contraddizione* $(A \to B) \to ((A \to \neg B) \to \neg A)$;

- il principio *ex falso sequitur quodlibet* $\neg A \rightarrow (A \rightarrow B)$.

L'intuizionismo si fonda sull'idea di presentare un oggetto matematico attraverso una prova costruttiva, cioè tramite una procedura algoritmica, anziché limitarsi ad affermare l'esistenza di tale oggetto. Di seguito, verrà illustrata la differenza tra una prova costruttiva e una prova non costruttiva per una determinata proposizione:

Example 1 (**Prova non costruttiva e prova costruttiva**). ***"Esistono due numeri a e b irrazionali, tali che*** a^{b} ***è razionale"***.

- ***Prova non costruttiva****: consideriamo* $\sqrt{2}^{\sqrt{2}}$*, può essere razionale o irrazionale. Se è razionale allora la proposizione è vera, se è irrazionale allora* $\left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = (\sqrt{2})^2 = 2$ *e la proposizione è dimostrata.*
- ***Prova costruttiva****: scelgo due numeri* $a = \sqrt{2}$*,* $b = \log_2(9)$*,* $a^b = 3$*.* $\sqrt{2}$*, è irrazionale e anche* $\log_2(9)$ *è irrazionale, in quanto se* $\log_2(9) = \frac{n}{m}$ *e quindi avremmo per le proprietà dei logaritmi che* $2^n = 9^m$ *ma il primo membro è pari, il secondo è dispari e quindi non possono essere uguali. Infine 3 è un numero razionale, per cui la proposizione è dimostrata.*

Dall'intuizionismo emerge la **logica intuizionista** [16], ideata da **Heyting** nel 1930. Questa forma di logica incorpora i principi basilari del ragionamento intuizionistico, fornendo le fondamenta per la logica della **matematica costruttiva** [17]. Successivamente, la logica è stata completamente sviluppata da **Gentzen** nel 1935 [18] e da **Kleene** nel 1952 [19], [20]. Inoltre, nel 1933 **Gödel** [21], [22] dimostrò l'equiconsistenza tra le teorie classiche e le teorie intuizioniste.
I sistemi intuizionisti hanno ispirato una varietà di interpretazioni, tra cui il tableaux di **Beth** [23] nel 1956 e i modelli topologici di **Rasiowa e Sikorski** [24] nel 1963. Successivamente **Kripke** [25] nel 1965 sviluppò un'interpretazione basata sui **modelli di Kripke**[1] la quale rispetto alla logica intuizionista del primo ordine risulta ***corretta e completa***, tuttavia, è importante notare che le dimostrazioni di completezza per la logica predicativa intuizionista richiedono l'impiego del ragionamento classico.
La logica intuizionista ha avuto molte applicazioni importanti nell'ambito dell'informatica come **l'isomorfismo di Curry-Howard** che collega le dimostrazioni intuizioniste ai lambda termini semplicemente tipati [26] e la **teoria costruttiva dei tipi di Per Martin-Löf**[27].

1.2 Sintassi IPL

La ragione per cui risulta necessario formalizzare la logica intuizionista è naturalmente motivata dalla spiegazione informale di Brouwer-Heyting-Kolmogorov della teoria intuizionista, delineata negli articoli sull'intuizionismo nella filosofia della matematica [9] e

[1] L'interpretazione fornita da Kripke sarà approfondita nelle sezioni successive in quanto risulta interessante per i nostri scopi.

nello sviluppo della logica intuizionista [28].
Il formalismo che descrive la logica intuizionista del primo ordine è denominato **H-IQC** Hilbertiano secondo Kleene (1952) ed è denominato spesso anche come **IFOL**. La sintassi del frammento intuizionista del primo ordine, è equivalente alla sintassi del frammento classico del primo ordine (FOL).

Definition 1 (Sintassi IPL). *La logica intuizionista proposizionale **H-IPC** denominata spesso anche come **IPL**, è il frammento della logica intuizionista del primo ordine **H-IQC** la cui sintassi è equivalente a quella del frammento proposizionale classico, che per completezza dell'elaborato è riportata di seguito:*

- *L'insieme delle formule atomiche $\mathcal{AP}$ è costituito dalle lettere proposizionali p,q,r... cioè predicati con arità 0;*

L'insieme delle formule ben formate $\mathcal{FO}$ è definito induttivamente come segue:

- *se $p \in \mathcal{AP}$ allora $p \in \mathcal{FO}$;*
- *se $\varphi \in \mathcal{FO}$ allora $\neg\varphi \in \mathcal{FO}$;*
- *se $\varphi, \psi \in \mathcal{FO}$, allora $(\varphi \vee \psi), (\varphi \wedge \psi), (\varphi \rightarrow \psi) \in \mathcal{FO}$.*

Prima di procedere alla sezione successiva riguardante la semantica IPL è fondamentale riportare un risultato molto importante dato dal seguente teorema:

Theorem 1 (Relazione tra formalismo classico e intuizionista). *Se nella lista degli assiomi riportata per la IPL, o equivalentemente per la IFOL, la **legge che esprime l'ex falso sequitur quodlibet** è sostituita con la classica **legge dell'eliminazione della doppia negazione**, o equivalentemente **la legge della contraddizione** è sostituita con il **principio del terzo escluso**, allora si ottiene un sistema formale classico proposizionale (PL), o equivalentemente un sistema classico del primo ordine (FOL).*

Questo risultato è ritenuto molto importante poiché indica che la logica intuizionista possiede la stessa consistenza della logica classica. Questo è stato dimostrato per la prima volta da **Glivenko** [29] nel 1929 per il frammento proposizionale, infatti egli ha dimostrato l'esistenza di una **traduzione** dalle teorie classiche a quelle intuizioniste per il frammento proposizionale e successivamente, nel 1933, questa traduzione è stata generalizzata per il frammento del primo ordine da **Gödel** e **Gentzen** [22] ed è nota come traduzione di **Gödel** - **Gentzen**. Gödel interpretò questi risultati dimostrando, nel suo lavoro [22], che le teorie intuizioniste si distinguono per la loro maggiore espressività rispetto alle teorie classiche.
In generale, la logica classica e la logica intuizionista, pur condividendo la stessa sintassi, sono in realtà due logiche diverse. Ad esempio, il "principio del terzo escluso", valido in logica classica, non è accettato in logica intuizionista, come sarà mostrato nella **proposizione 3**.
A partire dalla sezione successiva, prenderemo in considerazione solo il frammento proposizionale della logica intuizionista, poiché è sufficiente per gli scopi dell'elaborato.

1.3 Semantica IPL

Il metodo più diretto per dimostrare la validità di una formula F in un sistema formale S è quello di costruire una prova di F all'interno di S. Tuttavia, non è immediato riuscire a dimostrare se una formula è non valida in S. Spesso, l'incapacità di trovare una prova per F può suggerire la sua non validità, ma questa conclusione non è sempre giustificata. Ciò di cui spesso si ha bisogno è un'interpretazione rispetto alla quale S risulti:

- **Corretta**: ossia, se una formula F è provabile in S, allora risulterà valida in quell'interpretazione. Di conseguenza, per dimostrare che F non è provabile in S, è sufficiente dimostrare che non è valida in quell'interpretazione, tipicamente fornendo un contromodello per F.
- **Completa**: vale a dire che se una formula F è valida in quell'interpretazione, allora è provabile in S.

In questa sezione viene presentata una semantica sviluppata da **Saul Kripke** la quale risulta corretta e completa per la logica intuizionista.

Definition 2 (**Struttura di Kripke intuizionista**). *Una **struttura di Kripke intuizionista K** è costituita da una coppia $\langle S, \leq \rangle$ in cui S è un insieme numerabile (ovvero finito o nella cardinalità di $\mathbb{N}$) e $\leq \subseteq S \times S$ è un preordine su S, cioè una relazione binaria **riflessiva** e **transitiva**. Di seguito viene utilizzata la notazione $s' \geq s$ con il significato "s' è maggiore o uguale di s rispetto al preordine $\leq$".*

Definition 3 (**Modello di Kripke**). *Un **Modello di Kripke** $\mathcal{M}$ consiste in una tripla $\langle S, \leq, \mathcal{V} \rangle$ tale che $\langle S, \leq \rangle$ è una struttura di Kripke K e $\mathcal{V}$ è una funzione di valutazione monotonica $\mathcal{V} : S \to 2^{\mathcal{AP}}$, cioè tale che assegna ad ogni stato $s \in S$ un sottoinsieme di proposizioni atomiche e tale che per ogni coppia di stati $s, s' \in S$ si ha che se $s \leq s'$, allora $\mathcal{V}(s) \subseteq \mathcal{V}(s')$.*

Definition 4 (**Soddisfacibilità di una formula IPL**). *Dato un modello di Kripke $\mathcal{M}$, uno stato s e una formula φ, scriviamo $\mathcal{M}, s \models \varphi$ per "$\mathcal{M}$ **soddisfa** φ ad s". Tale relazione è definita induttivamente sulla struttura di φ come segue:*

- $\mathcal{M}, s \models p \in \mathcal{AP}$ *sse* $p \in \mathcal{V}(s)$;
- $\mathcal{M}, s \not\models \bot$ *per ogni stato s;*
- $\mathcal{M}, s \models \neg\varphi$ *sse per ogni stato* $s' \geq s$ *si ha che* $\mathcal{M}, s' \not\models \varphi$;
- $\mathcal{M}, s \models \varphi \wedge \psi$ *sse* $\mathcal{M}, s \models \varphi$ *e* $\mathcal{M}, s \models \psi$;
- $\mathcal{M}, s \models \varphi \vee \psi$ *sse* $\mathcal{M}, s \models \varphi$ *o* $\mathcal{M}, s \models \psi$;
- $\mathcal{M}, s \models \varphi \to \psi$ *sse per ogni stato* $s' \geq s$ *se* $\mathcal{M}, s' \models \varphi$ *allora* $\mathcal{M}, s' \models \psi$;

*Diremo che una formula φ è **valida** in un modello di Kripke $\mathcal{M}$ sse per ogni stato s di $\mathcal{M}$ si ha che $\mathcal{M}, s \models \varphi$.*
*Diremo che una formula φ è **valida** su una struttura di Kripke $K = \langle S, \leq \rangle$ sse per ogni valutazione monotonica $\mathcal{V}$ si ha che φ è soddisfatta nel modello $\mathcal{M} = \langle S, \leq, \mathcal{V} \rangle$.*
*Diremo che una formula φ è **valida** sse questa è soddisfatta su ogni struttura di Kripke K.*

Di seguito sono esposte alcune delle proprietà fondamentali della logica intuizionista proposizionale (IPL), evidenziando anche come alcune formule considerate valide in modo classico non lo siano nella logica intuizionista.

Proposition 1 (Relazione tra negazione ed implicazione). *Dato un modello di Kripke $\mathcal{M} = \langle S, \leq, \mathcal{V} \rangle$, per ogni stato s del modello, $\mathcal{M}, s \models (\varphi \to \bot) \iff \mathcal{M}, s \models \neg\varphi$*

Proof. ($\Longleftarrow$). Per ipotesi $\mathcal{M}, s \models \neg\varphi$. Allora per definizione questo significa che per ogni stato $s' \geq s$ $\mathcal{M}, s' \not\models \varphi$. Supponiamo per assurdo allora che $\mathcal{M}, s \not\models (\varphi \to \bot)$. Questo per definizione vuol dire che esiste uno stato $s'' \geq s$ tale che $\mathcal{M}, s'' \models \varphi$ ed $\mathcal{M}, s'' \not\models \bot$, ma dal momento che $s'' \geq s$, dall'ipotesi otteniamo che $\mathcal{M}, s'' \models \neg\varphi$ e allora $\mathcal{M}, s'' \models \varphi \wedge \neg\varphi$.
($\Longrightarrow$). Per ipotesi $\mathcal{M}, s \models (\varphi \to \bot)$. Per definizione questo vuol dire che per ogni stato $s' \geq s$ se $\mathcal{M}, s' \models \varphi$ allora $\mathcal{M}, s' \models \bot$. Ma dato che per definizione nessuno stato s può soddisfare $\bot$ vale necessariamente che $\mathcal{M}, s \models \neg\varphi$. □

Proposition 2 (Monotonia estesa). *Sia $\mathcal{M}$ un modello di Kripke, φ una formula ed s, s' una coppia di stati di $\mathcal{M}$. Se $\mathcal{M}, s \models \varphi$ ed $s \leq s'$ allora $\mathcal{M}, s' \models \varphi$.*

Proof. Questa proposizione si dimostra per induzione sulla struttura di φ:

- Caso base: $\varphi \equiv p \in \mathcal{AP}$. La proposizione è vero per definizione.
- Passo induttivo: si parte con l'ipotesi induttiva che se la proposizione è valida per tutte le formule φ con n connettivi allora lo sarà anche per tutte le formule φ con n+1 connettivi:
  - $\varphi \equiv \neg\psi$: $\mathcal{M}, s \models \neg\psi$ sse per ogni stato $s' \geq s$ $\mathcal{M}, s' \not\models \psi$. La proposizione è vera per definizione.
  - $\varphi \equiv \psi \vee \theta$: $\mathcal{M}, s \models \psi \vee \theta$ sse $\mathcal{M}, s \models \psi$ o $\mathcal{M}, s \models \theta$. Essendo che la proposizione risulta verificata per le formule ψ e θ per via dell'ipotesi induttiva, avremo che se s soddisfa almeno una tra ψ e θ allora la soddisferà necessariamente anche s' e di conseguenza s' soddisferà anche $\psi \vee \theta$.
  - $\varphi \equiv \psi \wedge \theta$: $\mathcal{M}, s \models \psi \wedge \theta$ sse $\mathcal{M}, s \models \psi$ e $\mathcal{M}, s \models \theta$. Essendo che la proposizione risulta verificata per le formule ψ e θ per via dell'ipotesi induttiva, avremo che se s soddisfa sia ψ che θ allora le soddisferà necessariamente anche s' e di conseguenza s' soddisferà anche $\psi \wedge \theta$.
  - $\varphi \equiv \psi \to \theta$. $\mathcal{M}, s \models \psi \to \theta$ sse per ogni $s' \geq s$ se $\mathcal{M}, s' \models \psi$ allora $\mathcal{M}, s' \models \theta$. Dato che per ipotesi induttiva la proposizione risulta verificata per ψ e θ e dato che $s \leq s'$ e $\mathcal{M}, s \models \psi \to \theta$ allora vale necessariamente che $\mathcal{M}, s' \models \psi \to \theta$.

□

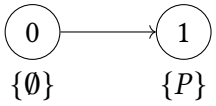
Proposition 3 (Formule intuizionisticamente non valide). *Le seguenti formule valide nella logica classica non risultano valide nella logica intuizionista:*

1. $P \vee \neg P$;
2. $(P \to Q) \longrightarrow (\neg P \vee Q)$;
3. $(P \to Q) \vee (Q \to P)$;
4. $\neg\neg P \to P$.

Proof. Per dimostrare la non validità intuizionista di queste formule è sufficiente trovare un **contromodello**, cioè un modello in cui almeno in un punto le formule non sono soddisfatte.

1. $P \vee \neg P$:

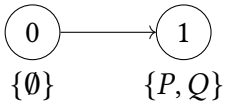
   $\mathcal{M}$:

   $\mathcal{M}, 0 \not\models P$ perché $P \notin \mathcal{V}(0) = \emptyset$.
   $\mathcal{M}, 0 \not\models \neg P$ perché $\mathcal{M}, 1 \models P$.

2. $(P \to Q) \longrightarrow (\neg P \vee Q)$;

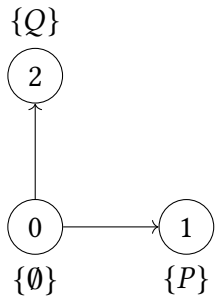
   $\mathcal{M}$:

   $\mathcal{M}, 0 \not\models P$, $\mathcal{M}, 0 \not\models Q$, inoltre $\mathcal{M}, 1 \models P \to Q$ e quindi $\mathcal{M}, 0 \models P \to Q$.
   $\mathcal{M}, 0 \not\models \neg P$ perché $\mathcal{M}, 1 \models P$

3. $(P \to Q) \vee (Q \to P)$;

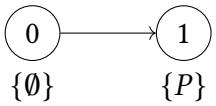
   $\mathcal{M}$:



$\mathcal{M}, 0 \not\models P \to Q$ perché $\mathcal{M}, 1 \not\models Q$.
$\mathcal{M}, 0 \not\models Q \to P$ perché $\mathcal{M}, 2 \not\models P$.

4. $\neg\neg P \to P$:

   $\mathcal{M}$:



   $\mathcal{M}, 0 \not\models P$ perché $P \notin \mathcal{V}(0) = \emptyset$.
   $\mathcal{M}, 0 \models \neg\neg P$ perché $\mathcal{M}, 1 \not\models \neg P$. [2]

□

Proposition 4 (Formule intuizionisticamente valide). *Le seguenti formule classicamente valide, restano valide anche intuizionisticamente:*

1. $P \to \neg\neg P$;
2. $(\neg P \vee Q) \longrightarrow (P \to Q)$.

Proof. consideriamo un generico modello $\mathcal{M}$ e un generico stato s:

1. $P \to \neg\neg P$.
   $\mathcal{M}, s \models P$. Per assurdo $\mathcal{M}, s \models \neg P$. Allora $\mathcal{M}, s \models P \wedge \neg P$.
2. $(\neg P \vee Q) \longrightarrow (P \to Q)$.
   Per ipotesi $\mathcal{M}, s \models \neg P \vee Q$. Per definizione allora $\mathcal{M}, s \models \neg P$ o $\mathcal{M}, s \models Q$.
   - Se $\mathcal{M}, s \models \neg P$ allora per ogni stato $s' \geq s$, $\mathcal{M}, s' \not\models P$ e dunque $\mathcal{M}, s' \models P \to Q$

[2] È interessante notare che il contromodello per mostrare che non è valido il principio di eliminazione della doppia negazione è lo stesso della non validità del principio del terzo escluso.

- $\mathcal{M}, s \models Q$ allora per ogni stato $s' \geq s$, $\mathcal{M}, s' \models Q$ e dunque $\mathcal{M}, s' \models P \rightarrow Q$

□

1.4 Logica intuizionista modale

La logica intuizionista può essere estesa con le modalità in differenti modi. Per ciascun frammento modale classico è definito un corrispondente intuizionista rimpiazzando la logica proposizionale (o del primo ordine) classica sottostante con la corrispondente intuizionista. **Plotkin** e **Stirling** [30] nel 1986 e successivamente **Simpson** [31] nel 1994 svilupparono dei framework per la logica modale intuizionista. L'idea generale è quella di considerare assiomi aggiuntivi per validare la traduzione della doppia negazione $\neg\neg$ di **Gödel-Gentzen** (**GG**) [32] anche nella logica modale.
La necessità di validare la traduzione anche per il frammento modale ha portato alla definizione di una nuova classe di modelli di Kripke detti **modelli birelazionali**[3].
Nella logica modale classica, comunemente nota come K, gli assiomi sono sviluppati esclusivamente per l'operatore $\Box$, fornendo successivamente una descrizione del comportamento dell'operatore $\Diamond$ [4]. Questi due operatori consentono rispettivamente di esprimere la "necessità" e la "possibilità" che una proprietà p sia vera. Tuttavia, nella logica intuizionista, la dualità tra le due modalità non è più valida. Di conseguenza, in questa prospettiva, l'attenzione si sposta verso una concezione più intrinseca delle modalità, dove entrambi gli operatori $\Box$ e $\Diamond$ assumono un ruolo fondamentale e devono essere trattati in modo distintivo senza poter essere derivati l'uno dall'altro, a differenza di come accade nella logica modale classica. Il problema di definire in modo ragionevole una logica modale intuizionista è stato oggetto di ricerca per decenni, e numerosi lavori sono stati dedicati a questo tema nel corso del tempo. Come prima logica modale intuizionista fu proposta **CK**, un'estensione della IPL con l'aggiunta dell'operatore $\Box$ e i due seguenti assiomi:

1. $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$;
2. $\Box(A \rightarrow B) \rightarrow (\Diamond A \rightarrow \Diamond B)$.

Nel 1948, **Fitch** [33] definì un modo per trattare l'operatore $\Diamond$ in contesto **CK**. Tuttavia, questo frammento fu abbandonato in quanto non permetteva di validare la traduzione di Gödel-Gentzen. Nel 1986 **Plotkin** introdusse il frammento **IK** [34], equivalente al frammento proposto in [35] nel 1984, il quale convalida la traduzione di Gödel-Gentzen. Tale frammento è un'estensione del frammento CK con l'aggiunta dei seguenti tre assiomi:

3. $\Diamond(A \vee B) \rightarrow (\Diamond A \vee \Diamond B)$;
4. $(\Diamond A \rightarrow \Box B) \rightarrow \Box(A \rightarrow B)$;

[3] Questi modelli in seguito verranno riportati e approfonditi in quanto risultano fondamentali per gli scopi dell'elaborato.

[4] i due operatori nella logica modale sono duali; infatti, $\Diamond p \equiv \neg\Box\neg p$

5. $\Diamond\bot \rightarrow \bot$.

Nel rispetto delle definizioni fornite dai modelli di Kripke per la logica modale intuizionista, è stato dimostrato che il frammento IK è sia **corretto** che **completo** [36], [37]. Inoltre, per alcuni sistemi modali intuizionisti, è stata anche dimostrata la **decidibilità** [31], [38].

–2–
Computation Tree Logic

Contents: **2.1 Logiche temporali. 2.2 Sintassi CTL. 2.3 Semantica CTL. 2.4 CTL-Model Checking.** Nel presente capitolo, viene fornita una panoramica sulle logiche temporali, procedendo

dall'introduzione dei concetti principali e offrendo una breve esplorazione delle proprietà temporali che è possibile descrivere in un modello. Successivamente, ci si concentrerà sull'approfondimento della logica CTL (Computation Tree Logic), fornendo una visione esaustiva dei suoi concetti fondamentali. In seguito, si procederà con l'analisi dettagliata della sintassi di CTL, esaminando ogni aspetto che compone questa struttura logica complessa. In seguito a ciò, ci si addentrerà nella comprensione della semantica sottostante a CTL, esplorando le interpretazioni dei suoi operatori modali e dei loro effetti sui modelli temporali presentando le principali proprietà accompagnate da **rigorose dimostrazioni svolte durante** la stesura dell'elaborato. Infine, verrà approfondita l'implementazione pratica della verifica delle formule CTL, fornendo un algoritmo di model checking efficiente.

2.1 Logiche temporali

I software e i sistemi software sono ormai pervasivi nella vita di tutti i giorni. Oltre alle tradizionali applicazioni come elaboratori di testo o fogli di calcolo eseguiti su workstation, il software ha assunto un ruolo fondamentale anche nei dispositivi a consumo, come gli smartphone, e nei sistemi complessi in cui viene utilizzato come dispositivo di controllo incorporato in automobili, aeroplani, centrali elettriche, impianti nucleari, impianti chimici e così via. In particolare, in questi ambiti applicativi, diventa essenziale garantire che il software implementato funzioni in modo corretto, sicuro e affidabile, poiché da esso può dipendere la vita umana. Ad esempio, il software all'interno del sistema di controllo dell'anti-slittamento di un'auto deve regolare con estrema precisione la velocità per stabilizzare l'auto, mentre per una centrale elettrica è di vitale importanza che nessun intruso possa prendere il controllo e che il sistema funzioni anche in caso di

guasto parziale di alcune delle sue componenti.
L'ingegneria del software ha il compito di fornire metodologie e standard per garantire la qualità del software. Tuttavia, nel contesto attuale, la legislazione e le autorità di certificazione richiedono una dimostrazione rigorosa e ben documentata delle proprietà più critiche del software tramite un processo di verifica accurato.
Negli ultimi anni, il concetto di Software as a Service (SaaS) ha introdotto un nuovo approccio nell'architettura dei sistemi software. Questi sistemi vengono sempre più considerati come entità autonome che agiscono in base a specifiche particolari. Tuttavia, la verifica di tali sistemi rappresenta una sfida complessa poiché il loro comportamento globale dipende fortemente dagli agenti coinvolti. Questo rende estremamente difficile analizzare tali sistemi prima della loro esecuzione. Un aspetto fondamentale della verifica consiste quindi nel verificare se ciascuna parte agisce in conformità alle specifiche che sono state definite per essa. In altre parole, è importante controllare se ogni componente si comporta come dovrebbe, secondo le regole e le funzionalità previste per essa.
Un possibile approccio consiste nell'arricchire la specifica delle proprietà del software mediante l'impiego delle **logiche temporali** [39], che costituiscono un insieme di formalismi destinati a ragionare e delineare le caratteristiche di sistemi in evoluzione nel tempo. Questi formalismi permettono di esprimere concetti intrinsecamente legati al tempo, come una sequenza temporale di eventi, consentendo così di modellare il sistema in base ad una successione di eventi e di considerare il tempo come una quantità discreta, ossia modellare vari i punti del sistema (stati) specificando degli eventi specifici che si verificano in essi, piuttosto che considerarlo come una grandezza continua. Di conseguenza, modellare gli eventi che si susseguono nel tempo all'interno di un sistema equivale a modellare una sua possibile computazione/esecuzione.
Attraverso le logiche temporali è possibile esprimere una vasta gamma di proprietà temporali, tra cui le seguenti risultano fondamentali:

- **invarianti**: una proprietà invariante è una proprietà che deve essere sempre verificata durante la computazione. È una condizione booleana;
- **safety**: proprietà indesiderate, ovvero situazioni che non devono mai verificarsi durante la computazione;
- **eventuality/liveness**: proprietà che prima o poi dovranno essere verificate durante la computazione;
- **proprietà di fairness**: queste proprietà assicurano che determinati eventi o azioni non vengano trascurati o bloccati in modo permanente durante l'esecuzione del sistema. A sua volta queste proprietà si suddividono principalmente in:
  - **unconditional fairness**: un evento deve essere schedulato infinite volte;
  - **weak fairness**: un evento che si verifica continuativamente prima o poi deve essere schedulato;
  - **strong fairness**: un evento che si verifica infinitamente spesso deve essere schedulato infinitamente spesso.

- **reachability**: tutte le proprietà descrivono le computazioni che si verificano all'interno del sistema, le quali sono individuabili come percorsi all'interno di un grafo indotto dal modello che si sta descrivendo. Pertanto la verifica di queste proprietà si riduce all'esplorazione di un grafo e in particolare, il problema di verificare se queste proprietà sono o non sono soddisfacibili nel modello è riformulabile in termini di (non) raggiungibilità di certi stati all'interno di quest'ultimo.

Le logiche temporali considerate nel capitolo attuale costituiscono un'estensione della logica proposizionale classica, pertanto gli operatori utilizzati per costruire formule temporali includono quelli classici, oltre a una serie di operatori temporali specifici mostrati di seguito.

Definition 5 (Operatori temporali). *Siano φ e ψ due formule. Le formule temporali sono costituite dagli operatori proposizionali classici più i seguenti operatori temporali:*

- *operatore **Next** (X), spesso indicato anche come "$\bigcirc$" che si applica ad una formula come $X\varphi$ e permette di esprimere il concetto "φ è vera nello stato **successivo**";*
- *operatore **Eventually** (F), spesso indicato anche come "$\Diamond$" che si applica ad una formula come $F\varphi$ e permette di esprimere il concetto "**prima o poi** φ sarà vera durante la computazione";*
- *operatore **Globally** (G) spesso indicato anche come "$\Box$" che si applica ad una formula come $G\varphi$ e permette di esprimere il concetto "φ è **sempre** vera durante la computazione";*
- *operatore **Until** ($\mathcal{U}$) che si applica a due formule come $\varphi \mathcal{U} \psi$ e permette di esprimere il concetto "φ è vera **finchè** ψ non diventa vera";*
- *operatore **Release** ($\mathcal{R}$) che si applica a due formule come $\varphi \mathcal{R} \psi$ e permette di esprimere il concetto "A partire dal momento in cui φ è vera, questa **rilascia** tutti i vincoli sulla soddisfazione di ψ";*

Le logiche temporali si suddividono in base al modello che si sta considerando, in particolare ci sono due tipologie di modello di tempo:

- **modello lineare**, dove vengono specificate le proprietà lungo una possibile computazione del modello in cui è possibile specificare una sequenza di istanti temporali. Per questo modello si utilizzano logiche come LTL (Linear Time Temporal Logic) [39] e PT-LTL (Past-Time LTL) [40];
- **modello branching**, dove il futuro non è univocamente determinato da una singola sequenza di istanti temporali, ma a partire da un certo punto si hanno a disposizione più possibili futuri che si possono dipartire. Questo modello è una generalizzazione del modello lineare e corrisponde ad un albero di percorsi. Per questo modello si utilizzano logiche come CTL (Computation Tree Logic) e CTL*.

È importante sottolineare come questi due modelli consentano di catturare differenti aspetti di un sistema. Nonostante il modello branching rappresenti una generalizzazione di quello lineare, ciò non implica automaticamente una maggior espressività temporale rispetto al modello lineare. Infatti, ci sono caratteristiche proprie del modello lineare che non possono essere adeguatamente espresse nel modello branching, e viceversa. Questo significa che non è necessariamente vero che una logica utilizzata in un modello sia più espressiva di un'altra: per esempio, esistono proprietà in LTL che non possono essere espresse in CTL, e viceversa [4].

2.2 Sintassi CTL

Prima di entrare nei dettagli della logica CTL e di comprendere appieno il suo funzionamento, è essenziale gettare le basi delineando il contesto in cui essa opera. Tale contesto è costituito dal modello su cui CTL si applica ed è noto come **albero di computazione**. Esplorare questo modello ci permette di comprendere meglio come le nozioni di tempo, stato e transizione sono modellate e interpretate, gettando le basi essenziali per una comprensione approfondita di CTL e dei suoi concetti associati.

Definition 6 (Struttura di Kripke). *Una struttura di Kripke è una tupla* $\mathcal{M} = \langle S, S_0, R, \mathcal{AP}, \mathcal{V} \rangle$ *tale che:*

- S *è un insieme di stati finito;*
- $S_0 \subseteq S$ *è un insieme di stati iniziali;*
- $R \subseteq S \times S$ *è una relazione di transizione binaria;*
- $\mathcal{AP}$ *è un insieme di proposizioni atomiche;*
- $\mathcal{V}$ *è una funzione di etichettatura del tipo* $\mathcal{V} : S \longrightarrow 2^{\mathcal{AP}}$ *tale che ad ogni stato associa un sottoinsieme di proposizioni atomiche.*

Definition 7 (Albero computazionale). *Sia* $\mathcal{M}$ *un modello di Kripke* $\mathcal{M} = \langle S, S_0, R, \mathcal{AP}, \mathcal{V} \rangle$ *e sia* s_0 *uno stato iniziale. La struttura* $TR(\mathcal{M}, s_0)$ *è detta* ***albero di computazione*** *di* $\mathcal{M}$ *con radice in* s_0 *ed è una tupla definita come segue:* $TR(\mathcal{M}, s_0) = \left\langle S_{s_0}, (s_0, \epsilon), \mathcal{R}_{s_0}, \mathcal{AP}, \mathcal{V}_{s_0} \right\rangle$ *tale che:*

- S_{s_0} *è un insieme di stati del tipo* (s, σ) *tale che s denota lo stato e* σ *denota il prefisso finito della stringa che termina in quello stato;*
- (s_0, ϵ) *denota lo stato iniziale (radice);*
- $\mathcal{AP}$ *è un insieme di proposizioni atomiche;*
- $\mathcal{R}_{s_0}$ *e* $\mathcal{V}_{s_0}$ *sono rispettivamente la relazione di transizione e la funzione di valutazione definite nel rispetto della seguente condizione:*
  Se $(s_1, \sigma) \in S_{s_0}$ *ed* $(s_1, s_2) \in \mathcal{R}$ *allora:*

i) $(s_2, \sigma.s_1) \in S_{s_0}$;

ii) $((s_1, \sigma), (s_2, \sigma.s_1)) \in \mathcal{R}_{s_0}$;

iii) $\mathcal{V}_{s_0}(s_1, \sigma) = \mathcal{V}(s_1)$.

Quindi dal modello di Kripke emerge un albero di computazione per ogni stato iniziale.

Dalla struttura definita risulta intuitivo che CTL utilizzerà gli operatori temporali definiti precedentemente ma aggiungendo dei simboli ad essi per indicare una quantificazione universale (**A**) o esistenziale (**E**) sui percorsi in cui deve valere la formula temporale specificata.

Definition 8 (Sintassi). *Dato un insieme di proposizioni atomiche $\mathcal{AP}$ l'insieme $\mathcal{FO}$ delle formule ben formate in CTL è definito induttivamente come segue:*

- *se $p \in \mathcal{AP}$ allora $p \in \mathcal{FO}$;*
- *se $\varphi \in \mathcal{FO}$, allora $\neg\varphi \in \mathcal{FO}$;*
- *se $\varphi, \psi \in \mathcal{FO}$ allora $\varphi \wedge \psi \in \mathcal{FO}$;*
- *se $\varphi, \psi \in \mathcal{FO}$ allora $\varphi \vee \psi \in \mathcal{FO}$;*
- *se $\varphi \in \mathcal{FO}$, allora $EX\varphi \in \mathcal{FO}$, questa formula esprime il concetto "**esiste** un percorso in cui è verificata $X\varphi$";*
- *se $\varphi, \psi \in \mathcal{FO}$ allora $E\varphi\mathcal{U}\psi \in \mathcal{FO}$, questa formula esprime il concetto "**esiste** un percorso in cui è verificata $\varphi\mathcal{U}\psi$";*
- *se $\varphi, \psi \in \mathcal{FO}$ allora $A\varphi\mathcal{U}\psi \in \mathcal{FO}$, questa formula esprime il concetto "**in ogni** percorso è verificata $\varphi\mathcal{U}\psi$".*

Fanno parte della sintassi CTL anche le costanti $\bot$ e $\top$ le quali indicano rispettivamente il simbolo dell'assurdo e della validità.

Definition 9 (Percorso). *Sia $\mathcal{M}$ una Kripke structure, ed s uno stato. Un **percorso** ρ che parte da s è una sequenza infinita di stati di $\mathcal{M}$ $\rho = s_0, s_1, s_2, .., s_n, ..$ tale che:*

- *$s_0 = s$;*
- *per ogni i si ha $R(s_i, s_{i+1})$.*

Si indica con ρ_i l'i+1-mo stato della sequenza.

2.3 Semantica CTL

Definition 10 (Soddisfacibilità di una formula CTL). *sia $\mathcal{M}$ una Kripke structure, s uno stato e φ una formula. Scriviamo $\mathcal{M}, s \models \varphi$ per "$\mathcal{M}$ soddisfa φ ad s" o equivalentemente $TR(\mathcal{M}, s), (s, \epsilon) \models \varphi$ per "l'albero di computazione radicato in s soddisfa φ". Tale relazione è definita induttivamente sulla struttura di φ come segue:*

- $\mathcal{M}, s \models p \in \mathcal{AP}$ *sse* $p \in \mathcal{V}(s)$;
- $\mathcal{M}, s \models \neg\varphi$ *sse* $\mathcal{M}, s \not\models \varphi$;
- $\mathcal{M}, s \models \varphi \vee \psi$ *sse* $\mathcal{M}, s \models \varphi$ *oppure* $\mathcal{M}, s \models \psi$;
- $\mathcal{M}, s \models \varphi \wedge \psi$ *sse* $\mathcal{M}, s \models \varphi$ *e* $\mathcal{M}, s \models \psi$;
- $\mathcal{M}, s \models EX\varphi$ *sse esiste uno stato* s' *tale che* $R(s, s')$ *e* $\mathcal{M}, s' \models \varphi$;
- $\mathcal{M}, s \models E\varphi\mathcal{U}\psi$ *sse esiste un percorso* ρ *che parte da s* $(\rho_0 = s)$ *tale che* $\mathcal{M}, \rho_i \models \psi$ *per un* $i \geq 0$ *e per ogni* $0 \leq j < i$ $\mathcal{M}, \rho_j \models \varphi$;
- $\mathcal{M}, s \models A\varphi\mathcal{U}\psi$ *sse per ogni percorso* ρ *che parte da s* $(\rho_0 = s)$ $\mathcal{M}, \rho_i \models \psi$ *per un* $i \geq 0$ *e per ogni* $0 \leq j < i$ $\mathcal{M}, \rho_j \models \varphi$.

Diremo che $\mathcal{M} \models \varphi$ che sta per "$\mathcal{M}$ è un modello di φ" sse per ogni $s \in S_0$ $\mathcal{M}, s \models \varphi$. Infine diremo che una formula è ***CTL-valida*** *sse è vera in ogni modello $\mathcal{M}$.*

In CTL alcuni operatori possono essere derivati dalle formule $\mathcal{FO}$, e la loro equivalenza costituisce delle formule CTL-valide.

Definition 11 (Operatori derivati). *Definiamo $\varphi \rightarrow \psi$ come $\neg\varphi \vee \psi$ e $\varphi \leftrightarrow \psi$ come $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$. Le seguenti formule sono CTL-valide:*

- $AX\psi \leftrightarrow \neg EX\neg\psi$;
- $EF\psi \leftrightarrow E\top\mathcal{U}\psi$;
- $AF\psi \leftrightarrow A\top\mathcal{U}\psi$;
- $AG\psi \leftrightarrow \neg EF\neg\psi$;
- $EG\psi \leftrightarrow \neg AF\neg\psi$;
- $E\varphi\mathcal{R}\psi \leftrightarrow \neg A\neg\varphi\mathcal{U}\neg\psi$;
- $A\varphi\mathcal{R}\psi \leftrightarrow \neg E\neg\varphi\mathcal{U}\neg\psi$.

Di seguito viene riportata la semantica degli operatori derivati per assicurare la completezza e un maggiore approfondimento dell'elaborato.

Definition 12 (Semantica degli operatori derivati in CTL). *Dato un modello $\mathcal{M}$ ed uno stato s, la semantica degli operatori derivati è definita come segue:*

- $\mathcal{M}, s \models AX\psi$ *sse per ogni stato* s' *tale che* $R(s, s')$, $\mathcal{M}, s' \models \psi$;
- $\mathcal{M}, s \models EF\psi$ *sse esiste un percorso* ρ *che parte da s* ($\rho_0 = s$) *tale che* $\rho_i \models \psi$ *per un* $i \geq 0$;
- $\mathcal{M}, s \models AF\psi$ *sse per ogni percorso* ρ *che parte da s* ($\rho_0 = s$) $\rho_i \models \psi$ *per un* $i \geq 0$;
- $\mathcal{M}, s \models EG\psi$ *sse esiste un percorso* ρ *che parte da s* ($\rho_0 = s$) *tale che* $\rho_i \models \psi$ *per ogni* $i \geq 0$;
- $\mathcal{M}, s \models AG\psi$ *sse per ogni percorso* ρ *che parte da s* ($\rho_0 = s$) $\rho_i \models \psi$ *per ogni* $i \geq 0$;
- $\mathcal{M}, s \models E\varphi\mathcal{R}\psi$ *sse esiste un percorso* ρ *che parte da s* ($\rho_0 = s$) *tale che per ogni* $i \geq 0$ ($\rho_i \models \psi$ *oppure per un* $0 \leq i \leq j$, $\rho_j \models \varphi$);
- $\mathcal{M}, s \models A\varphi\mathcal{R}\psi$ *sse per ogni percorso* ρ *che parte da s* ($\rho_0 = s$) *si ha che per ogni* $i \geq 0$ ($\rho_i \models \psi$ *oppure per un* $0 \leq i \leq j$, $\rho_j \models \varphi$).

Per maggiore chiarezza è riportata di seguito una visualizzazione grafica della semantica delle formule:

Example 2 (Formule soddisfatte nei modelli). *Consideriamo come modello $\mathcal{M}$ il seguente albero:*

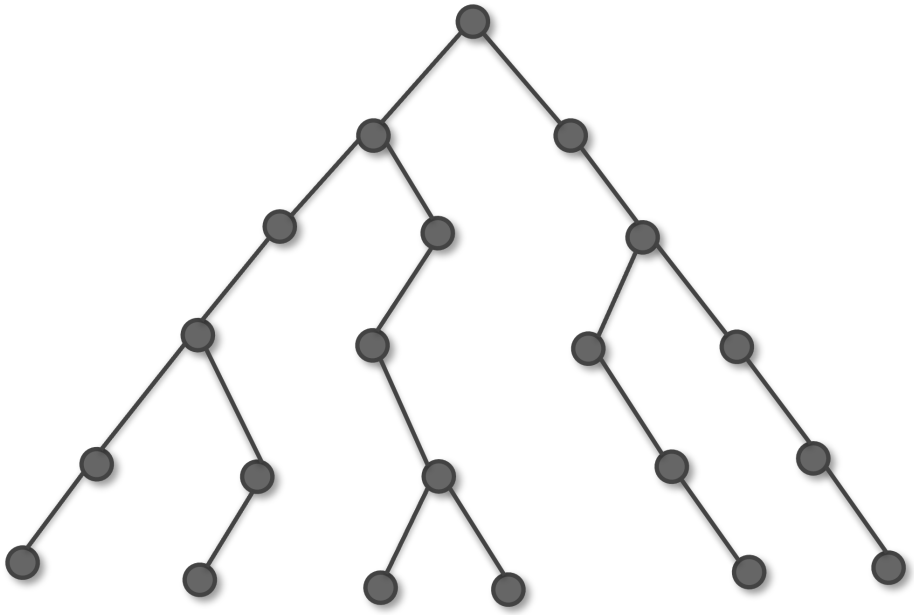


Figure 2.1: Albero di computazione $\mathcal{M}$.

In questo modello si assume che la radice sia uno stato iniziale, pertanto la soddisfazione delle formule temporali nel modello dovrà essere verificata solo a partire da quest'ultima.

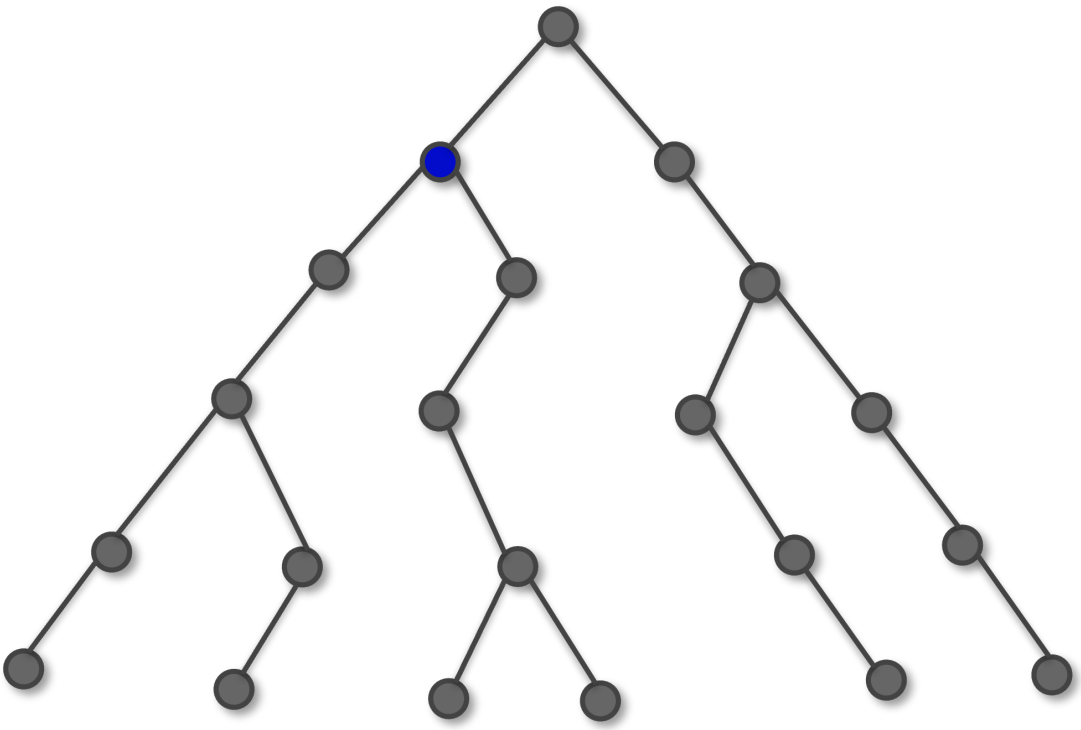


Figure 2.2: $\mathcal{M} \models EX\psi$.

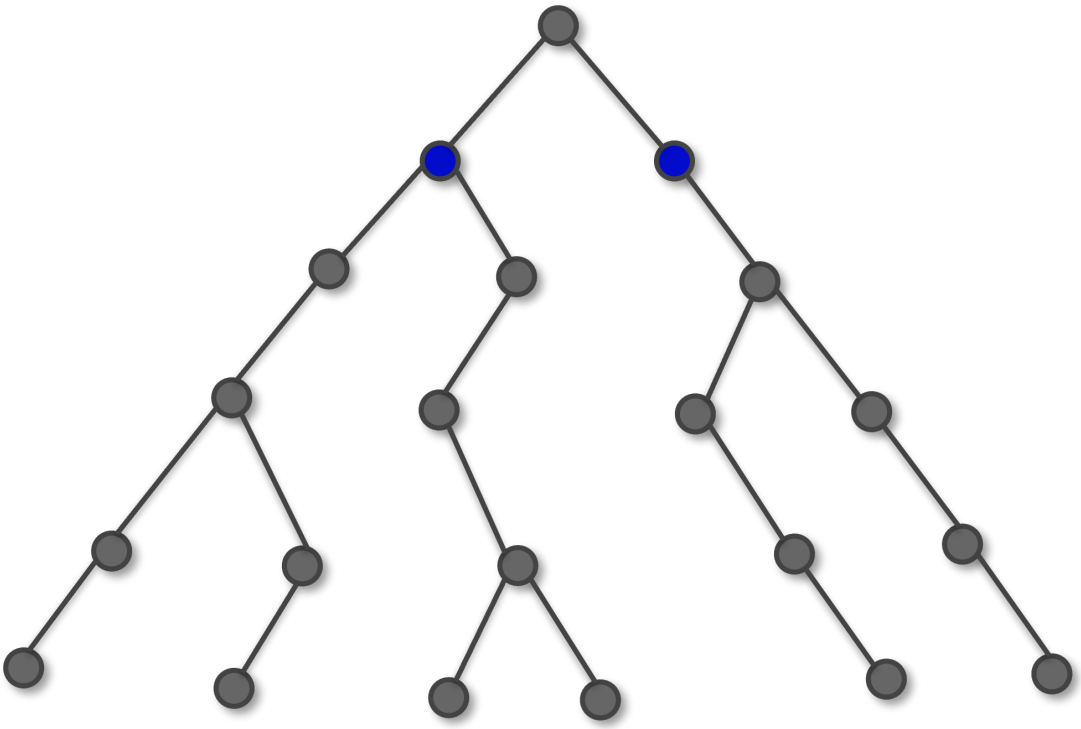


Figure 2.3: $\mathcal{M} \models AX\psi$.

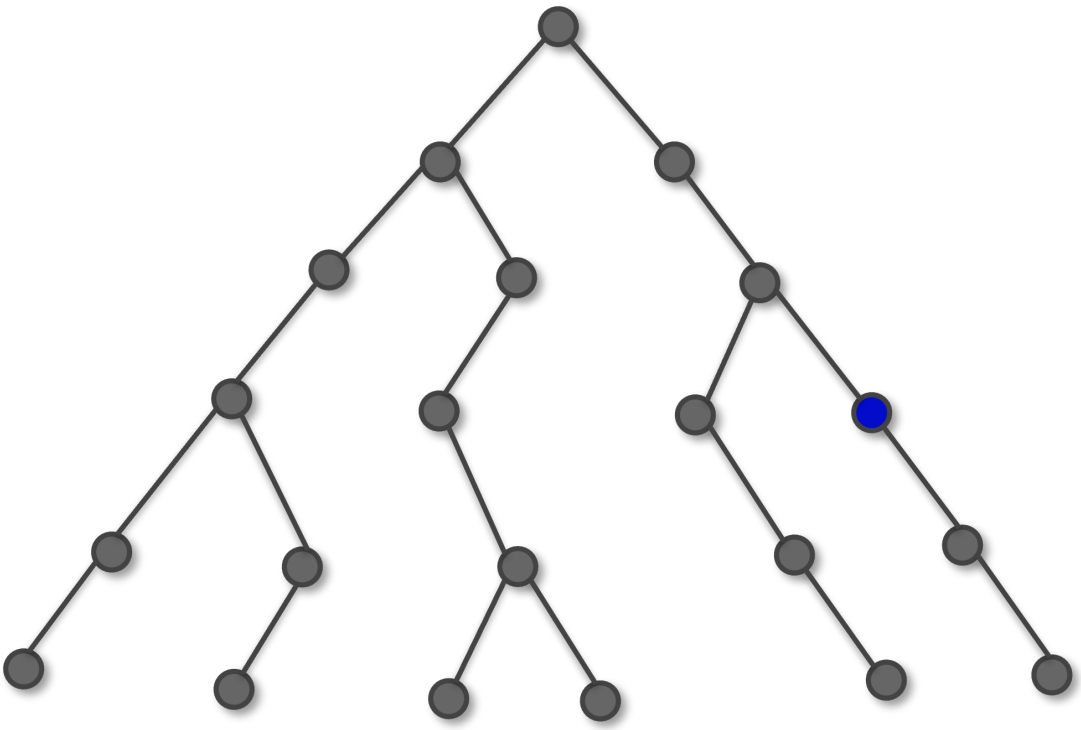


Figure 2.4: $\mathcal{M} \models EF\psi$.

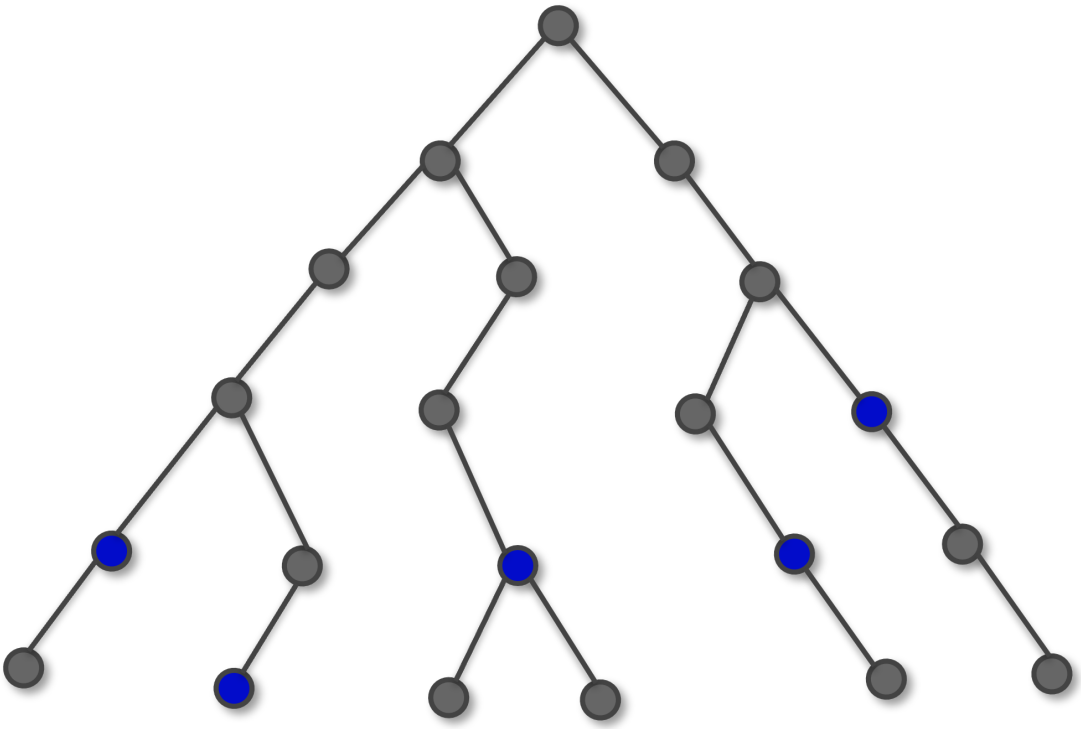


Figure 2.5: $\mathcal{M} \models AF\psi$.

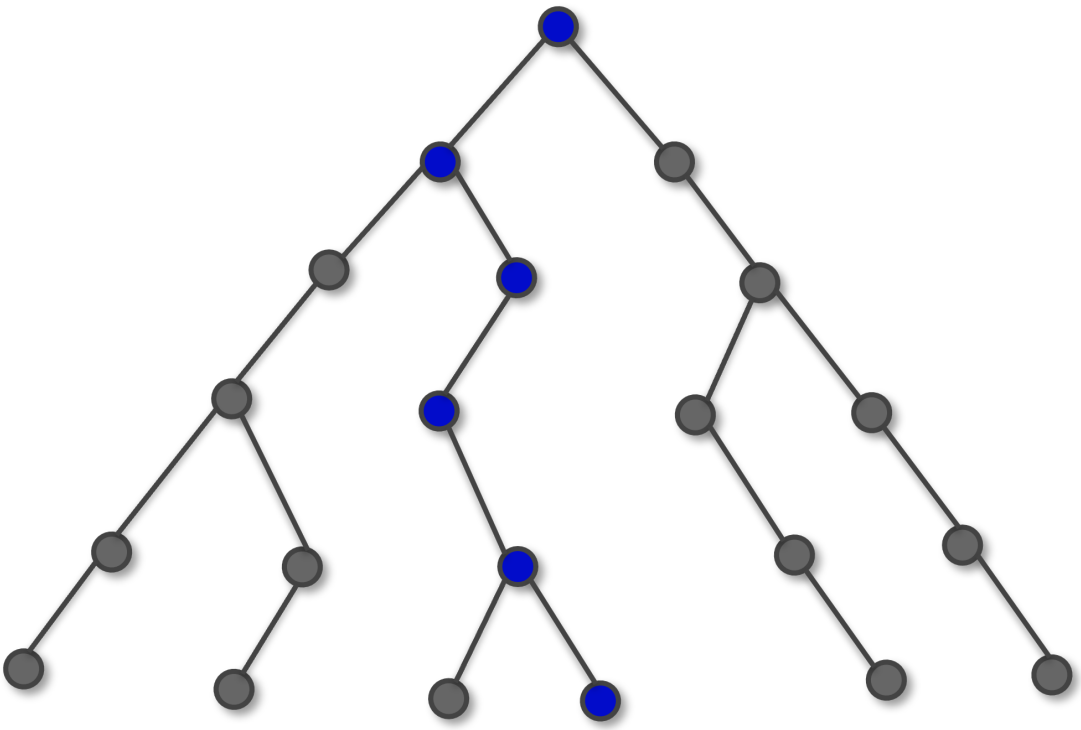


Figure 2.6: $\mathcal{M} \models EG\psi$.

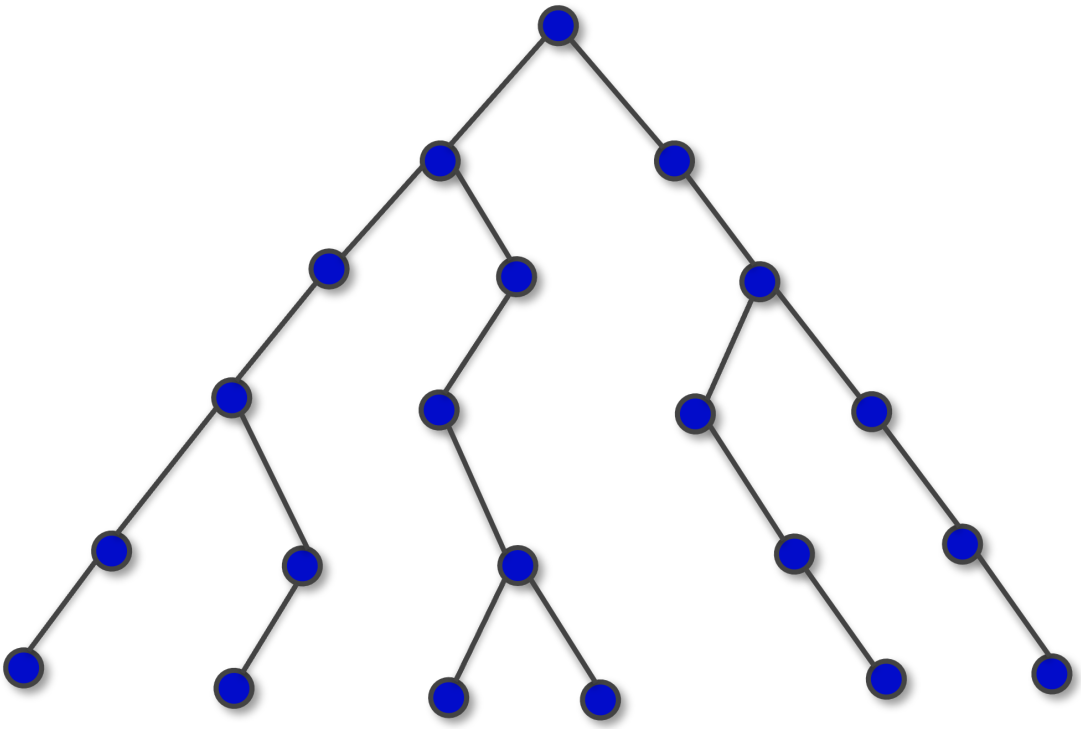


Figure 2.7: $\mathcal{M} \models AG\psi$.

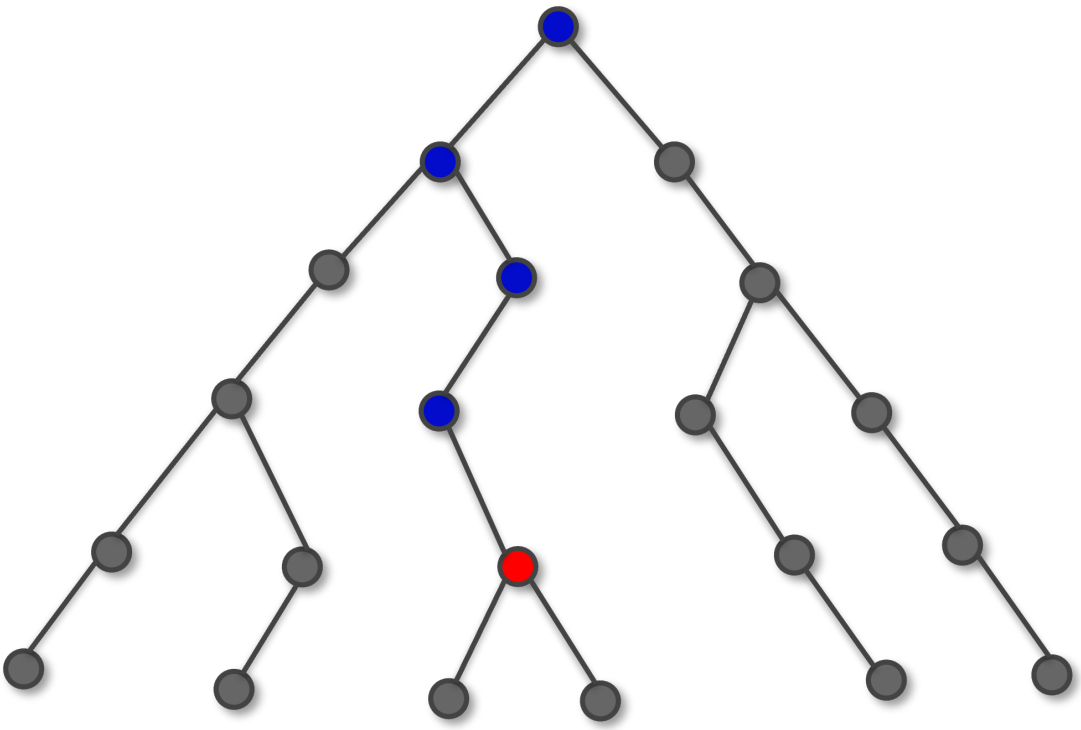


Figure 2.8: $\mathcal{M} \models E\varphi\mathcal{U}\psi$.

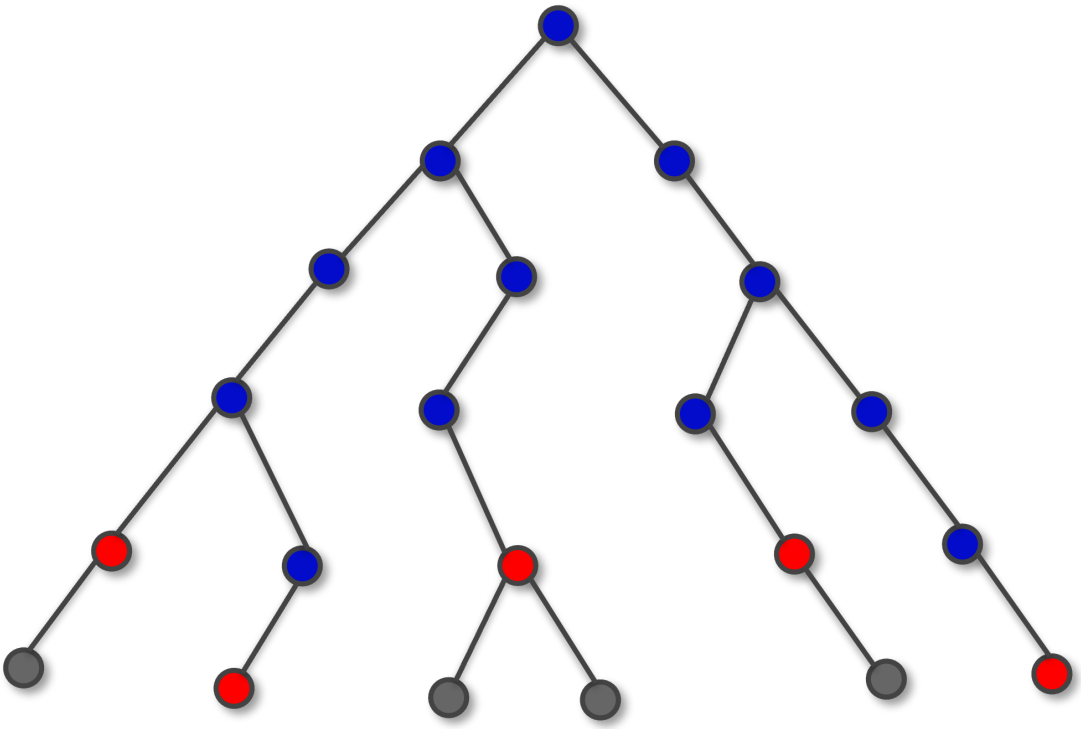


Figure 2.9: $\mathcal{M} \models A\varphi\mathcal{U}\psi$.

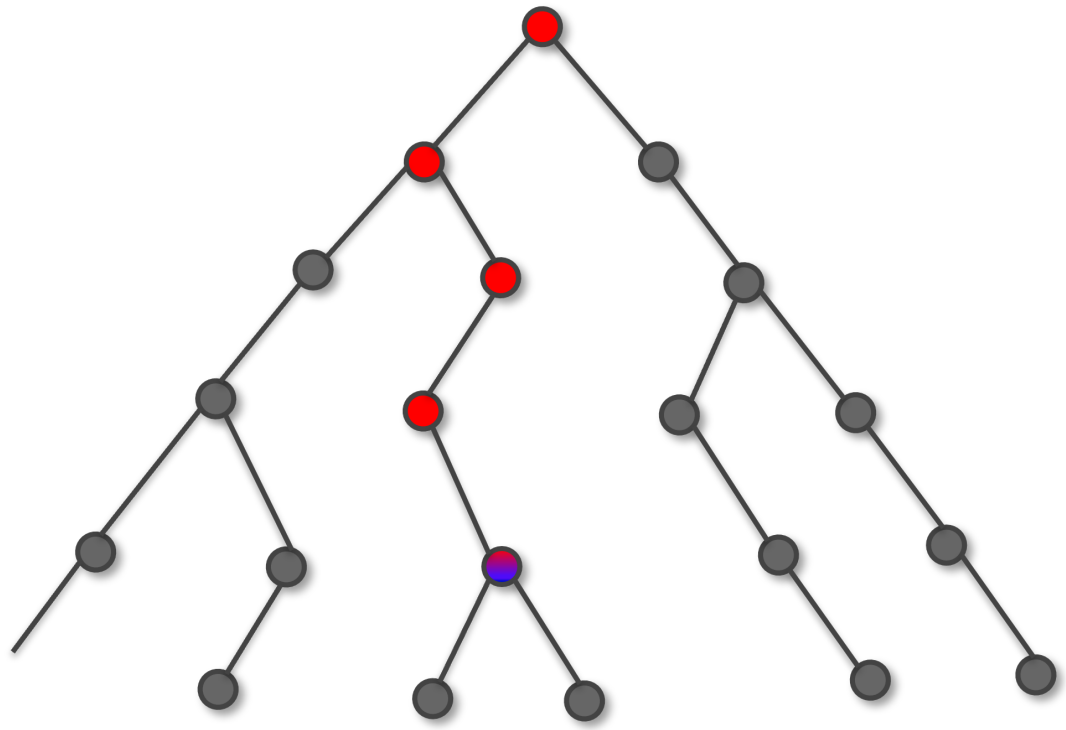


Figure 2.10: $\mathcal{M} \models E\varphi\mathcal{R}\psi$.

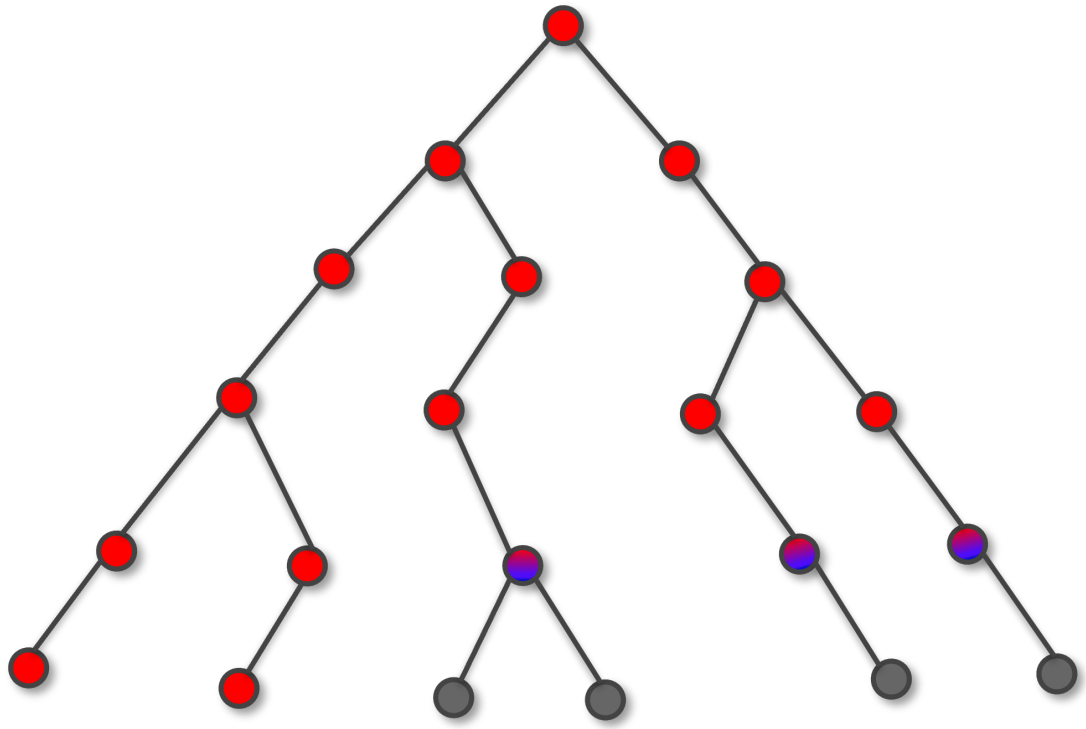


Figure 2.11: $\mathcal{M} \models A\varphi\mathcal{R}\psi$.

*Il doppio colore negli stati in **figura 2.10** e in **figura 2.11** indica che in quei punti sono soddisfatte entrambe φ e ψ.*

Un altro importante risultato sulla validità delle formule è riportato nella seguente proposizione:

Proposition 5 (Relazione tra A$\varphi\mathcal{U}\psi$ ed EGφ). *Le seguenti formule sono CTL-valide:*

1. $EG\psi \leftrightarrow \neg A\top\mathcal{U}\neg\psi$;
2. $A\varphi\mathcal{U}\psi \leftrightarrow \neg EG\neg\psi \wedge \neg E\neg\psi\mathcal{U}(\neg\varphi \wedge \neg\psi)$.

Proof. Dimostriamo prima (1) e poi (2):

- (1) ($\rightarrow$): vero perchè $EG\psi \leftrightarrow \neg AF\neg\psi \leftrightarrow \neg A\top\mathcal{U}\neg\psi$;
  ($\leftarrow$): $\mathcal{M}, s \models \neg A\top\mathcal{U}\neg\psi$ sse esiste un percorso ρ che parte da s ($\rho_0 = s$) tale che (1) $M, \rho_i \models \psi$ per ogni $i \geq 0$ oppure (2) esiste $0 \leq j < i$ tale che $\mathcal{M}, \rho_j \not\models \top$). la (2) è sempre falsa in ogni stato soddisfa $\top$ per definizione. Allora si ottiene che $\mathcal{M}, s \models \neg A\top\mathcal{U}\neg\psi$ sse $M, \rho_i \models \psi$ per ogni $i \geq 0$ e questa è la definizione di $\mathcal{M}, s \models EG\psi$.

- (2) ($\rightarrow$): chiaramente se $\mathcal{M}, s \models A\varphi\mathcal{U}\psi$ allora $\mathcal{M}, s \models \neg EG\neg\psi$.
  Supponiamo per assurdo che $\mathcal{M}, s \not\models \neg E\neg\psi\mathcal{U}(\neg\varphi\wedge\neg\psi)$. Allora $\mathcal{M}, s \models E\neg\psi\mathcal{U}(\neg\varphi\wedge\neg\psi)$ e questo per definizione significa che esiste un percorso ρ che parte da s ($\rho_0 = s$) tale che $\mathcal{M}, \rho_i \models \neg\varphi \wedge \neg\psi$ per un $i \geq 0$ e per ogni $0 \leq j < i$ $\mathcal{M}, \rho_j \models \neg\psi$. Per ipotesi $\mathcal{M}, s \models A\varphi\mathcal{U}\psi$, pertanto affinchè entrambe le due formule siano soddisfatte deve necessariamente esistere un $i' > i$ tale che $\mathcal{M}\rho_{i'} \models \psi$ e per ogni $0 \leq j < i'$ $\mathcal{M}, \rho_j \models \varphi$. Ma per $j = i$ si ha che $\mathcal{M}, \rho_j \models \varphi \wedge \neg\varphi$ e questo è un assurdo.
  ($\leftarrow$) Per assurdo $\mathcal{M}, s \not\models A\varphi\mathcal{U}\psi$. Per definizione vuol dire che esiste un percorso ρ che parte da s (ρ_0) $= s$ tale che per ogni $i \geq 0$ $\mathcal{M}, \rho_i \models \neg\psi$ (1), oppure $\mathcal{M}, s \models \neg\varphi$ per un $0 \leq j < i$ (2). Per ipotesi $\mathcal{M}, s \models \neg EG\neg\psi$ e quindi $\mathcal{M}, \int \models \mathcal{AF}\psi$. quindi in ogni percorso che si diparte da s deve esistere un punto in cui è verificata ψ, per cui la (1) non può valere. Allora l'unica soluzione è avere un percorso ρ che parte da s ($\rho_0 = s$) in cui $\mathcal{M}, \rho_i \models \psi$ per un $i \geq 0$ e $\mathcal{M}, \rho_j \models \neg\varphi$ per ogni $0 \leq j < i$. Allora scegliamo il più piccolo i tale per cui $\mathcal{M}, \rho_i \models \psi$. Allora per ogni punto precedente (incluso ρ_j) si ha che è soddisfatta $\neg\psi$. Allora $\mathcal{M}, s \models E\neg\psi\mathcal{U}(\neg\varphi \wedge \neg\psi)$ e questo è un assurdo.

□

Questo risultato ci permette di sostituire nella sintassi una formula del tipo $A\varphi\mathcal{U}\psi$ con una formula del tipo $EG\varphi$ [4],[3],[5] e quindi di semplificare il processo di verifica delle formule poiché gli algoritmi come il model checking analizzeranno solo le formule che quantificano esistenzialmente sui percorsi, considerando la verifica delle formule che quantificano universalmente sui percorsi, come dei casi aggiuntivi. Prima di descrivere altre proprietà di CTL occorre dare le seguenti definizioni:

Definition 13 (**Pre-immagine esistenziale**). *Sia $\mathcal{M}$ un modello e Q un insieme di stati di $\mathcal{M}$. L'insieme $pre_\exists(Q)$ è detto pre-immagine esistenziale dell'insieme Q ed è definito come segue: $pre_\exists(Q) = \{q \in S |\ \exists q' \text{ t.c. } R(q,q') \text{ e } q' \in Q\}$. cioè è l'insieme di tutti gli stati di $\mathcal{M}$ tali che hanno un arco uscente in almeno uno stato dell'insieme Q.*

Definition 14 (**Pre-immagine universale**). *Sia $\mathcal{M}$ un modello e Q un insieme di stati di $\mathcal{M}$. L'insieme $pre_\forall(Q)$ è detto pre-immagine universale dell'insieme Q ed è definito come segue: $pre_\forall(Q) = \{q \in S |\ \forall q'\ R(q,q') \Rightarrow q' \in Q\}$. cioè è l'insieme di tutti gli stati di $\mathcal{M}$ tali che hanno tutti gli archi uscenti in almeno uno stato dell'insieme Q.*

Per dimostrare alcune proprietà nell'ambito CTL, è possibile fare riferimento alla notazione di linguaggio di una formula. Questo approccio mira a semplificare l'esposizione dei risultati delle prove all'interno del contesto CTL, rispetto alla tradizionale semantica precedentemente definita. L'utilizzo di tale notazione linguistica offre un'opportunità per delineare in modo più dettagliato e comprensibile le caratteristiche e le dimostrazioni all'interno della logica, consentendo una maggiore chiarezza nell'analisi condotta all'interno di questo ambito specifico della logica computazionale.

Definition 15 (**CTL in termini di linguaggio**). *Dato un modello $\mathcal{M}$ e una formula φ si denota l'insieme degli stati di $\mathcal{M}$ che soddisfano φ come $[\![\varphi]\!]^{\mathcal{M}} = \{w \in W | \mathcal{M}, w \models \varphi\}$. se $\mathcal{M}$ è contestualmente dato allora è possibile ometterlo nella definizione e scrivere soltanto $[\![\varphi]\!]$. Infine se Y è un insieme di stati, si denota con Y^c il suo complemento.*

Proposition 6 (**Relazione tra formule esistenziali e pre immagini**). *Dato un modello $\mathcal{M}$, ed una formula φ. Allora si ha che $[\![EX\varphi]\!] = Pre_\exists([\![\varphi]\!])$.*

Proof. Dimostriamo entrambi i lati:
($\longrightarrow$). consideriamo uno stato s tale che $\mathcal{M}, s \models EX\varphi$. Per definizione questo vuol dire che esiste uno stato s' tale che $R(s,s')$ ed $\mathcal{M}, s' \models \varphi$. Allora per definizione $s \in pre_\exists(\varphi)$.
($\longleftarrow$). consideriamo uno stato s tale che $s \in Pre_\exists([\![\varphi]\!])$. Questo vuol dire che esiste uno stato s' tale che $R(s,s')$ ed $s' \in [\![\varphi]\!]$. Questa è la definizione per $\mathcal{M}, s \models EX\varphi$. □

Proposition 7 (**Relazione tra formule universali e pre immagini**). *Dato un modello $\mathcal{M}$, ed una formula φ. Allora si ha che $[\![AX\varphi]\!] = Pre_\forall([\![\varphi]\!])$*

Proof. Dimostriamo entrambi i lati:
($\longrightarrow$). consideriamo uno stato s tale che $\mathcal{M}, s \models AX\varphi$. Per definizione questo vuol dire che per ogni stato s' tale che $R(s,s')$, $\mathcal{M}, s' \models \varphi$. Allora per definizione $s \in pre_\forall(\varphi)$.
($\longleftarrow$). consideriamo uno stato s tale che $s \in Pre_\forall([\![\varphi]\!])$. Questo vuol dire che per ogni stato s' tale che $R(s,s')$ allora $s' \in [\![\varphi]\!]$. Questa è la definizione per $\mathcal{M}, s \models AX\varphi$. □

Definition 16 (**Punto fisso**). *Sia X un insieme ed f una funzione definita come $f : 2^X \longrightarrow 2^X$. un **punto fisso** per f è un sottoinsieme $Y \subseteq X$ tale che $f(Y) = Y$.*

Definition 17 (**Funzione monotona**). *Sia X un insieme ed f una funzione definita come $f : 2^X \longrightarrow 2^X$. Si dice che f è monotona se e solo se per ogni sottoinsieme $Y, Z \subseteq X$ si ottiene che se $Y \subseteq Z$ allora $f(Y) \subseteq f(Z)$.*

Il teorema di **Knaster-Tarski** [7] afferma che se f è una funzione monotona allora essa ammette un più piccolo e un più grande punto fisso. Cioè vale a dire:

- esiste un punto fisso Y' tale che $Y' \subseteq Z$ per ogni punto fisso Z;
- esiste un punto fisso Y'' tale che $Z \subseteq Y''$ per ogni punto fisso Z.

Verrà mostrato in seguito che l'esistenza di due punti fissi per una funzione monotona f definita sulle parti di un insieme risulta essenziale per definire un algoritmo di model checking per CTL efficiente.

Proposition 8 (**Equivalenze del punto fisso**). *Le seguenti formule sono CTL-valide:*

1. $EF\varphi \leftrightarrow \varphi \vee EXEF\varphi$;
2. $EG\varphi \leftrightarrow \varphi \wedge EXEG\varphi$;
3. $E\varphi_1 \mathcal{U} \varphi_2 \leftrightarrow \varphi_2 \vee (\varphi_1 \wedge EXE\varphi_1 \mathcal{U} \varphi_2)$;
4. $E\varphi_1 \mathcal{R} \varphi_2 \leftrightarrow \varphi_2 \wedge (\varphi_1 \vee EXE\varphi_1 \mathcal{R} \varphi_2)$;
5. $AF\varphi \leftrightarrow \varphi \vee AXAF\varphi$;
6. $AG\varphi \leftrightarrow \varphi \wedge AXAG\varphi$;
7. $A\varphi_1 \mathcal{U} \varphi_2 \leftrightarrow \varphi_2 \vee (\varphi_1 \wedge AXA\varphi_1 \mathcal{U} \varphi_2)$;
8. $A\varphi_1 \mathcal{R} \varphi_2 \leftrightarrow \varphi_2 \wedge (\varphi_1 \vee AXA\varphi_1 \mathcal{R} \varphi_2)$.

Proof. Dimostriamo (2), (3) in quanto risultano fondamentali per gli scopi dell'elaborato. Le dimostrazioni rimanenti saranno interamente simili.

2. ($\longrightarrow$).
   Supponiamo che $\mathcal{M}, s \models EG\varphi$. Questo significa che esiste un percorso ρ che parte da s ($\rho_0 = s$) tale che $\mathcal{M}, \rho_i \models \varphi$ per ogni $i \geq 0$. Questo implica, in particolare, che per $\rho_0 = s$ abbiamo $\mathcal{M}, s \models \varphi$. Inoltre, per ogni $j \geq 1$, $\mathcal{M}, \rho_j \models \varphi$, quindi $\mathcal{M}, \rho_0 \models EXEG\varphi$ come desiderato.
   ($\longleftarrow$). Supponiamo che $\mathcal{M}, s \models \varphi \wedge EXEG\varphi$. Questo significa che esiste un percorso ρ che parte da s ($\rho_0 = s$) tale che $\mathcal{M}, \rho_1 \models EG\varphi$. Dalla semantica di CTL otteniamo che esiste un percorso τ che parte da ρ_1 ($\tau_0 = \rho_1$) e $\mathcal{M}, \tau_i \models \varphi$ per ogni $i \geq 0$. Consideriamo quindi il percorso $\pi = s \cdot \tau$. Questo percorso soddisfa $G\varphi$ per definizione. Pertanto, otteniamo che $\mathcal{M}, s \models EG\phi$.

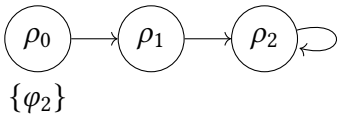
3. ($\longrightarrow$) Supponiamo che $\mathcal{M}, s \models E\varphi_1 \mathcal{U} \varphi_2$. Questo significa che esiste un percorso ρ che parte da s ($\rho_0 = s$) tale che $\mathcal{M}, \rho_i \models \varphi_2$ per un $i \geq 0$ e $\mathcal{M}, \rho_j \models \varphi$ per ogni $1 \leq j < i$. Se $i = 0$, otteniamo che $\mathcal{M}, s \models \varphi_2$, quindi l'equivalenza è dimostrata, altrimenti, si ha che $\mathcal{M}, s \models \varphi_1$. Consideriamo ora il suffisso $\rho_{\geq 1}$ di ρ cioè il suffisso

che indica il percorso che parte da ρ_1. Questo percorso soddisfa $\varphi_1 \mathcal{U} \varphi_2$. Quindi esiste un percorso ρ che parte da s ($\rho_0 = s$) che soddisfa $\varphi_1 \mathcal{U} \varphi_2$ a partire dalla sua seconda posizione, cioè ρ_1 ed è proprio $\rho_{\geq 1}$. Di conseguenza, $\mathcal{M}, s \models \varphi_1 \wedge EXE\varphi_1 \mathcal{U} \varphi_2$ come desiderato.
($\longleftarrow$). Supponiamo che $\mathcal{M}, s \models \varphi_2 \vee (\varphi_1 \wedge \text{EXE}\varphi_1 \mathcal{U} \varphi_2)$. Nel caso in cui $\mathcal{M}, s \models \varphi_2$ risulta immediato che $\mathcal{M}, s \models \varphi_1 \mathcal{U} \varphi_2$. Nel caso in cui $\mathcal{M}, s \models (\varphi_1 \wedge \text{EXE}\varphi_1 \mathcal{U} \varphi_2)$, questo per definizione significa che esiste un percorso ρ che parte da s ($\rho_0 = s$) tale per cui $\mathcal{M}, \rho_1 \models E\varphi_1 \mathcal{U} \varphi_2$. Dalla semantica CTL otteniamo che esiste un percorso τ che parte da ρ_1 ($\tau_0 = \rho_1$) e $\mathcal{M}, \tau_i \models \varphi_2$ per un $i \geq 0$ e che per ogni $0 \leq j < i$ $\mathcal{M}, \tau_j \models \varphi_1$. Consideriamo quindi il percorso $\pi = s \cdot \tau$. Questo percorso soddisfa $\varphi_1 \mathcal{U} \varphi_2$ per definizione. Pertanto, otteniamo che $\mathcal{M}, s \models E\varphi_1 \mathcal{U} \varphi_2$.

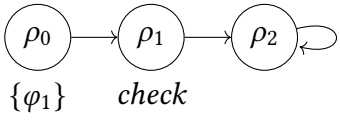
□

Le equivalenze del punto fisso indicano che i percorsi che soddisfano le specifiche CTL possono essere costruiti step-by-step. Inoltre, le soluzioni al problema di verifica, così come il controllo della soddisfacibilità, possono essere ottenute attraverso una procedura iterativa. La visione algoritmica del calcolo degli stati che soddisfano una formula temporale può essere formalizzata attraverso logiche di calcolo del punto fisso, cioè varianti del μ-calcolo modale.

Example 3 (Analisi di una equivalenza del punto fisso). *Consideriamo la formula $E\varphi_1 \mathcal{U} \varphi_2 \leftrightarrow \varphi_2 \vee (\varphi_1 \wedge EXE\varphi_1 \mathcal{U} \varphi_2)$. Per verificare se il modello soddisfa $E\varphi_1 \mathcal{U} \varphi_2$ è possibile ragionare step-by-step sugli stati del modello anzichè considerare l'intera sequenza temporale. Consideriamo il percorso $\rho = \rho_0, \rho_1, ..., \rho_n, ...$ se $\mathcal{M}, \rho_0 \models \varphi_2$ allora $E\varphi_1 \mathcal{U} \varphi_2$ è soddisfatta per definizione.*



Se $\mathcal{M}, \rho_0 \not\models \varphi_2$ e $\mathcal{M}, \rho_0 \not\models \varphi_1$, allora chiaramente $E\varphi_1 \mathcal{U} \varphi_2$ non è soddisfatta in $\mathcal{M}, \rho_0$. Ma se $\mathcal{M}, \rho_0 \models \varphi_1$, allora è possibile iterare sul percorso ρ andando a verificare nello stato successivo, cioè in ρ_1, quali sono le formule soddisfatte, equivalentemente a come è stato verificato in ρ_0.



Questo controllo "check" corrisponde a chiedersi se in ρ_1, cioè nello stato successivo a quello che è stato appena controllato nel percorso ρ (operatore EX), è soddisfatta $E\varphi_1 \mathcal{U} \varphi_2$ controllando se questo stato soddisfa φ_2 oppure soddisfa φ_1 e bisogna continuare ad iterare la procedura su tutto il percorso ρ.

Prima di passare alla sezione successiva, è fondamentale menzionare un'altra proprietà che collega gli operatori temporali all'esistenza di un massimo e un minimo punto fisso di una funzione monotona. In particolare, bisogna enunciare e dimostrare il seguente teorema [8]:

Theorem 2. *Sia $\mathcal{M}$ un modello di Kripke e φ, ψ due formule. Allora si ha che:*

1. $[\![EG\psi]\!]$ *è il più grande punto fisso della funzione* EG_ψ *dove* $EG_\psi(X) = [\![\psi]\!] \cap pre_\exists(X)$;
2. $[\![E\varphi \mathcal{U} \psi]\!]$ *è il più piccolo punto fisso della funzione* $E\mathcal{U}_{\varphi,\psi}$ *dove* $E\mathcal{U}_{\varphi,\psi}(X) = [\![\psi]\!] \cup ([\![\varphi]\!] \cap pre_\exists(X))$;
3. $[\![E\varphi \mathcal{R} \psi]\!]$ *è il più grande punto fisso della funzione* $E\mathcal{R}_{\varphi,\psi}$ *dove* $E\mathcal{R}_{\varphi,\psi}(X) = [\![\psi]\!] \cap ([\![\varphi]\!] \cup pre_\exists(X))$;
4. $[\![AG\psi]\!]$ *è il più grande punto fisso della funzione* AG_ψ *dove* $AG_\psi(X) = [\![\psi]\!] \cap pre_\forall(X)$;
5. $[\![A\varphi \mathcal{U} \psi]\!]$ *è il più piccolo punto fisso della funzione* $A\mathcal{U}_{\varphi,\psi}$ *dove* $E\mathcal{U}_{\varphi,\psi}(X) = [\![\psi]\!] \cup ([\![\varphi]\!] \cap pre_\forall(X))$;
6. $[\![A\varphi \mathcal{R} \psi]\!]$ *è il più grande punto fisso della funzione* $A\mathcal{R}_{\varphi,\psi}$ *dove* $E\mathcal{R}_{\varphi,\psi}(X) = [\![\psi]\!] \cap ([\![\varphi]\!] \cup pre_\forall(X))$.

Proof. Dimostriamo i punti (1), (2) e (6) poiché i restanti sono del tutto simili:

1. Bisogna dimostrare che $X = [\![EG\psi]\!]$ è un punto fisso della funzione EG_ψ (1) e che è il più grande (2).
   (1). Per la **proposizione 8** si ha che $X = [\![EG\psi]\!] = [\![\psi \wedge EXEG\psi]\!]$. Allora $X = [\![\psi]\!] \cap [\![EXEG\psi]\!]$ e per la **proposizione 6** si conclude che $X = [\![\psi]\!] \cap Pre_\exists([\![EG\psi]\!])$.
   (2). Per dimostrare che X è il massimo punto fisso della funzione EG_ψ, bisogna dimostrare che considerando un ulteriore punto fisso Y, se uno stato $s_0 \in Y$ allora $s_0 \in X$. Consideriamo un $s_0 \in Y$, per definizione $s_0 \in [\![\psi]\!] \cap Pre_\exists(Y)$ ed esiste un s_1 tale che $R(s_0, s_1)$. Allora anche $s_1 \in Y$, pertanto è possibile costruire induttivamente un percorso del tipo $s_0, s_1, s_2, .., s_n, ..$ tale che per ogni $i \geq 0$ $s_i \in Y$ e di conseguenza $s_i \in [\![\psi]\!]$. Allora per la definizione di $X = [\![EG\psi]\!]$ si ottiene che $s_0 \in X$.

2. Bisogna dimostrare che $X = [\![E\varphi \mathcal{U} \psi]\!]$ è un punto fisso della funzione $EU_{\varphi,\psi}$ (1) e che è il più piccolo (2).
   (1). Per la **proposizione 8** si ha che $X = [\![E\varphi \mathcal{U} \psi]\!] = [\![\psi \vee (\varphi \wedge EXE\varphi \mathcal{U} \psi)]\!]$. Allora $X = [\![\psi]\!] \cup ([\![\varphi]\!] \cap [\![EXE\varphi \mathcal{U} \psi]\!])$ e per la **proposizione 6** si conclude che $X = [\![\psi]\!] \cup ([\![\varphi]\!] \cap Pre_\exists([\![\varphi \mathcal{U} \psi]\!]))$.
   (2). Per dimostrare che X è il minimo punto fisso della funzione $EU_{\varphi,\psi}$, bisogna dimostrare che considerando un ulteriore punto fisso Y, se $s \in X$ allora $s \in Y$. Per definizione di X esiste un percorso ρ che parte da s ($\rho_0 = s$) ed un $n \geq 0$ tale che

$\rho_n \in [\![\psi]\!]$ e per ogni $0 \leq i < n$ si ha che $\rho_i \in [\![\varphi]\!]$, e dato che $Y = [\![\psi]\!] \cup ([\![\varphi]\!] \cap Pre_\exists(Y))$ allora $\rho_n \in Y$. Bisogna dimostrare ora che se $\rho_i \in Y$ allora $\rho_{i-1} \in Y$ per concludere che $s \in Y$. Per ipotesi $\rho_i \in Y$ e dato che ci troviamo all'interno di un percorso si ha che $R(\rho_{i-1}, \rho_i)$. Pertanto $\rho_{i-1} \in Pre_\exists(Y)$ e dato che $\rho_{i-1} \in [\![\varphi]\!]$ allora $\rho_{i-1} \in ([\![\varphi]\!] \cap Pre_\exists(Y)) \subseteq Y$. Allora si conclude che $\rho_{i-1} \in Y$ e quindi anche $s \in Y$.

6. Bisogna dimostrare che $X = [\![A\varphi \mathcal{R} \psi]\!]$ è un punto fisso della funzione $ER_{\varphi,\psi}$ (1) e che è il più grande (2).
(1). Per la **proposizione 8** si ha che $X = [\![A\varphi \mathcal{R} \psi]\!] = [\![\psi \wedge (\varphi \vee AXA\varphi \mathcal{U} \psi)]\!]$. Allora $X = [\![\psi]\!] \cap ([\![\varphi]\!] \cup [\![AXA\varphi \mathcal{R} \psi]\!])$ e per la **proposizione 7** si conclude che $X = [\![\psi]\!] \cap ([\![\varphi]\!] \cup Pre_\forall([\![\varphi \mathcal{R} \psi]\!]))$.
(2). Per dimostrare che X è il massimo punto fisso della funzione $AR_{\varphi,\psi}$, bisogna dimostrare che considerando un ulteriore punto fisso Y, se uno stato $s_0 \in Y$ allora $s_0 \in X$. Per definizione, dato che Y è un punto fisso, $s_0 \in [\![\psi]\!] \cap ([\![\varphi]\!] \cup Pre_\forall(Y))$. Consideriamo un percorso ρ che parte da s_0 ($\rho_0 = s_0$) e dal momento che $s_0 \in [\![\psi]\!]$ consideriamo l'insieme $\chi = \{k \geq 1 : \rho_k \notin Y\}$. se $\chi = \emptyset$, allora ogni $\rho_i \in [\![\psi]\!]$ e quindi è possibile concludere. Altrimenti sia $j \geq 1$ il più piccolo intero tale per cui $\rho_j \notin Y$. Supponiamo per assurdo che $\rho_{j-1} \notin [\![\varphi]\!]$. Allora dato che $\rho_{j-1} \in Y$ e $\rho_j \notin [\![\varphi]\!]$ vale necessariamente che $\rho_j \in [\![\psi]\!] \cap Pre_\forall(Y)$. Allora per ogni percorso π che parte da ρ_{j-1} ($\pi_0 = \rho_{j-1}$) si ottiene che $\pi_1 = \rho_j \in Y$ e questo è un assurdo.

□

2.4 CTL-Model Checking

I principali task di verifica si suddividono in:

- **Satisfiability** (SAT), cioè data una formula φ l'obiettivo è quello di trovare un'assegnazione oppure un modello $\mathcal{M}$ tale che $\mathcal{M} \models \varphi$;
- **Validity**, cioè data una formula φ l'obiettivo è quello di verificare se questa è valida in tutti i modelli della logica presa in considerazione;
- **Model Checking** (MC), cioè data una formula φ e dato un modello $\mathcal{M}$ l'obiettivo è quello di rispondere alla domanda "$\mathcal{M} \models \varphi$?".
- **Module Checking**, cioè data una formula φ, dato un modulo o modello $\mathcal{M}$, l'obiettivo è quello di verificare se $\mathcal{M} \models \varphi$ in tutte le possibili interazioni che il sistema può fare con l'ambiente.

I task di Validity e SAT, non vengono comunemente applicati in un contesto pratico per due motivi fondamentali. Il primo motivo è strettamente legato al fatto che, nella pratica, l'approccio standard prevede la costruzione di un modello unico. Questo modello deve soddisfare una serie di proprietà per garantire il rispetto dei requisiti software. Di

conseguenza, è piuttosto insolito iniziare con una formula per costruire il modello. Inoltre, sebbene la conoscenza di formule valide o soddisfacibili possa contribuire alla costruzione di un modello più efficace, la loro applicazione pratica è limitata. Nonostante ciò, sia Validity che SAT rimangono degli strumenti teorici fondamentali per comprendere a pieno i principi sottostanti alla logica e alla teoria dei calcoli, fornendo una base solida per l'analisi e la progettazione di sistemi complessi.
Il secondo motivo è strettamente legato alla complessità computazionale. Infatti, un problema di Model Checking risulta quasi sempre più gestibile dal punto di vista computazionale rispetto ad altri task. In particolare, il Model Checking è un problema intrinsecamente completo: dato un modello $\mathcal{M}$ e una formula φ, l'algoritmo determina sempre se $\mathcal{M} \models \varphi$, oppure fornisce un controesempio nel caso in cui φ non è soddisfatta. Questa caratteristica non si riscontra in altri task come SAT. Infatti, in alcune logiche, come nella logica del primo ordine o in Strategy Logic (SL), SAT risulta essere un problema indecidibile, pertanto risulta un task computazionalmente proibitivo in un contesto pratico.
Il module checking invece risulta un problema molto più complesso del model checking in termini computazionali. Inoltre, si pone su una prospettiva diversa rispetto al model checking in quanto può essere più adatto per la verifica dei sistemi aperti, cioè, sistemi che devono adattare il loro comportamento all'input che ricevono dall'ambiente. Quindi, nonostante sia un metodo affascinante, il model checking risulta più idoneo per la verifica di proprietà di correttezza per uno specifico modello. In generale la scelta di un task di verifica dipende in gran parte dal tipo di sistema che si sta cercando di verificare.
L'idea del model checking CTL è quella di andare a verificare quali formule sono vere in ciascuno stato, e dato che la semantica delle formule è definita induttivamente allora la soddisfazione di una formula dipenderà dalla soddisfazione delle sue sottoformule.

Definition 18 (Sottoformule di φ in CTL). *Sia φ una formula. L'insieme delle sottoformule di φ che si denota con $SF(\varphi)$ è definito induttivamente come segue:*

- $\varphi \in SF(\varphi)$;
- *se* $\neg\psi \in SF(\varphi)$ *allora* $\psi \in SF(\varphi)$;
- *se* $\psi_1 \vee \psi_2 \in SF(\varphi)$ *allora* $\psi_1, \psi_2 \in SF(\varphi)$;
- *se* $\psi_1 \wedge \psi_2 \in SF(\varphi)$ *allora* $\psi_1, \psi_2 \in SF(\varphi)$;
- *se* $EX\psi \in SF(\varphi)$ *allora* $\psi \in SF(\varphi)$;
- *se* $EG\psi \in SF(\varphi)$ *allora* $\psi \in SF(\varphi)$;
- *se* $E\psi_1 \mathcal{U} \psi_2 \in SF(\varphi)$ *allora* $\psi_1, \psi_2 \in SF(\varphi)$.

Dalla definizione risulta che il numero di sottoformule di una formula φ è lineare nella lunghezza di φ: $|SF(\varphi)| = \theta(|\varphi|)$.
La versione standard dell'algoritmo di CTL model checking combina l'idea di calcolare una formula a partire dalle sue sottoformule con il calcolo del punto fisso. Cioè la verifica

delle proprietà temporali al "passo successivo" viene fatta calcolando l'appropriata pre-immagine, mentre la verifica delle proprietà temporali "a lungo termine" viene fatta calcolandone il corrispondente punto fisso.
L'algoritmo riceve in input un modello $\mathcal{M}$ e una formula φ. Egli Procede ricorsivamente sulle sottoformule di φ, trattando ogni tipo di sottoformula in modo specifico. L'algoritmo termina restituendo tutti gli stati che soddisfano φ. Se nessuno stato soddisfa la formula, viene restituito l'insieme vuoto $\emptyset$. Inoltre sono riportati anche i casi addizionali per il controllo delle formule che quantificano universalmente sui percorsi.

Theorem 3 (Classe di complessità del CTL model checking). *L'algoritmo di CTL model checking è **PTIME-COMPLETE**.*

Proof. Bisogna provare:

- **Hardness**: Per verificare una formula temporale l'algoritmo deve passare attraverso degli stati lungo un percorso. Risulta chiaro allora che la verifica delle proprietà temporali è riconducibile ad un problema di **reachability**, cioè uno stato (non) soddisfa una formula sse questo (non) è raggiungibile a partire da uno stato iniziale. Dato che reachability è PTIME-COMPLETE ed esiste una riduzione da questo problema al CTL model checking allora la hardness è dimostrata.

- **Completeness**: per verificare che l'algoritmo si trova effettivamente in PTIME è necessario fare un'analisi dell'algoritmo. L'algoritmo effettua una chiamata ricorsiva per ogni sottoformula di φ per un totale di $|SF(\varphi)|$ chiamate. Dobbiamo dimostrare che ogni chiamata ricorsiva impiega al più tempo $O(|R|)$. Nel caso di una proposizione atomica e dei connettivi booleani è abbastanza chiaro che sia così. Per quanto riguarda il calcolo delle pre-immagini (sia esistenziale che universale) per un dato insieme Q può essere fatto al più in tempo $O(|R|)$ e infine, il while richiede al più $O(|S|)$ iterazioni. In generale il tempo impiegato dal while è al più $O(|R|)$ in quanto il calcolo delle pre immagini può essere implementato in modo tale che ogni transizione del modello venga visitata una e una sola volta durante le varie iterazioni del ciclo. Pertanto l'algoritmo si trova in **PTIME**.

□

Corollary 1 (Complessità computazionale del CTL model checking). *L'algoritmo di model checking CTL può essere risolto in tempo $O(|\varphi| * |R|)$ quindi in tempo polinomiale sull'input.*

Contrariamente ad altre logiche temporali, il CTL model checking si distingue per la sua efficienza, emergendo come uno degli algoritmi di verifica più rapidi. Infatti il model checking per logiche temporali come LTL o CTL* è un algoritmo classificato come PSPACE-COMPLETE, indicando una complessità computazionale gestibile, ma comunque maggiore.

Algorithm 1 mCheckCTL

Require: $\mathcal{M}, \varphi$
Ensure: $\{s|\ \mathcal{M}, s \models \varphi\}$

switch φ **do**
  case $\varphi \equiv p$ return $\{s|p \in \mathcal{V}(s)\}$;
  case $\varphi \equiv \neg\psi$: return $S \setminus mCheckCTL(\mathcal{M}, \psi)$;
  case $\varphi \equiv \psi_1 \wedge \psi_2$: return $mCheckCTL(\mathcal{M}, \psi_1) \cap mCheckCTL(\mathcal{M}, \psi_2)$;
  case $\varphi \equiv \psi_1 \vee \psi_2$: return $mCheckCTL(\mathcal{M}, \psi_1) \cup mCheckCTL(\mathcal{M}, \psi_2)$;
  case $\varphi \equiv E\mathcal{X}\psi$: return $pre_\exists(mCheckCTL(\mathcal{M}, \psi))$;
  case $\varphi \equiv EG\psi$:
    $Q_1 := S$;
    $Q_2 := mCheckCTL(\mathcal{M}, \psi)$;
    while $Q_1 \not\subseteq Q_2$ **do**
      $Q_1 := Q_2$;
      $Q_2 := pre_\exists(Q_1) \cap Q_1$;
    end while
    return Q_1;
  case $\varphi \equiv E\psi_1\mathcal{U}\psi_2$:
    $Q_1 := \emptyset$;
    $Q_2 := mCheckCTL(\mathcal{M}, \psi_1)$;
    $Q_3 := mCheckCTL(\mathcal{M}, \psi_2)$;
    while $Q_3 \not\subseteq Q_1$ **do**
      $Q_1 := Q_1 \cup Q_3$;
      $Q_3 := pre_\exists(Q_1) \cap (Q_2)$;
    end while
    return Q_1;
  case $\varphi \equiv A\mathcal{X}\psi$: return $pre_\forall(mCheckCTL(\mathcal{M}, \psi))$;
  case $\varphi \equiv AG\psi$:
    $Q_1 := S$;
    $Q_2 := mCheckCTL(\mathcal{M}, \psi)$;
    while $Q_1 \not\subseteq Q_2$ **do**
      $Q_1 := Q_2$;
      $Q_2 := pre_\forall(Q_1) \cap Q_1$;
    end while
    return Q_1;
  case $\varphi \equiv A\psi_1\mathcal{U}\psi_2$:
    $Q_1 := \emptyset$;
    $Q_2 := mCheckCTL(\mathcal{M}, \psi_1)$;
    $Q_3 := mCheckCTL(\mathcal{M}, \psi_2)$;
    while $Q_3 \not\subseteq Q_1$ **do**
      $Q_1 := Q_1 \cup Q_3$;
      $Q_3 := pre_\forall(Q_1) \cap (Q_2)$;
    end while
    return Q_1;

–3–
Intuitionistic Computation Tree Logic

Contents: **3.1 Intuizionismo e CTL. 3.2 Sintassi ICTL. 3.3 Semantica ICTL. 3.4 Differenze tra CTL e ICTL. 3.5 Semantica ICTL sotto una nuova condizione. 3.6 ICTL-Model Checking.**

Nei capitoli precedenti è stata esaminata approfonditamente la logica intuizionista, partendo dal fornire dei cenni storici e confrontandola poi con l'approccio classico. Successivamente è stata esibita una semantica corretta e completa ed è stato illustrato il suo funzionamento in relazione alle principali proprietà tipiche dell'intuizionismo e infine sono stati riportati i principi base della logica intuizionista modale. Successivamente sono state introdotte le logiche temporali, discutendo delle varie proprietà esprimibili e approfondendo in particolare la logica CTL, presentando la sua sintassi e semantica e illustrando poi alcune delle sue proprietà fondamentali come le equivalenze del punto fisso, accompagnate da un efficace algoritmo di model checking. In questo capitolo, ci si concentra sulla combinazione delle due logiche esibite, che permette di definire una versione intuizionista di CTL, chiamata **ICTL**. Come nei capitoli precedenti, inizieremo esplorando la sintassi della logica, seguita dalla definizione dei modelli birelazionali che consentono di definire una semantica adeguata, rispettando i principi sia della logica intuizionista che della logica CTL. Questo ci permetterà di esporre anche le principali proprietà che caratterizzano questa logica.
A tal fine, introdurremo gli enunciati presentati nel paper [13], di cui sono state **completate** le dimostrazioni all'interno dell'elaborato. Inoltre, **è stata sviluppata** una seconda versione di ICTL sotto una nuova condizione, poiché quella proposta in [13] presentava alcune limitazioni come la **non** validità delle equivalenze del punto fisso per gli operatori temporali universali. Infine, per questa nuova versione di ICTL, verrà presentato un algoritmo di model checking efficiente, completando così la prima parte dell'elaborato.

3.1 Intuizionismo e CTL

La combinazione della logica intuizionista e CTL si propone di formalizzare la seguente idea: voler distinguere tra due diverse evoluzioni temporali all'interno di un modello CTL. Una rappresenta la conoscenza dell'agente sul sistema, mentre l'altra rappresenta le possibili evoluzioni del sistema stesso basate sulla conoscenza data. L'obiettivo dell'agente è verificare in modo costruttivo che determinate proprietà siano soddisfatte rispetto alle possibili evoluzioni del modello e all'evoluzione potenziale della sua conoscenza. In altre parole, data una certa proprietà φ, si desidera verificare che, in qualsiasi stato della conoscenza dell'agente riguardo al modello e indipendentemente dall'evoluzione del modello stesso, questa sia soddisfatta. Questo tipo di intuizione potrebbe permettere di considerare la verifica di proprietà CTL in nuovi contesti, e ne sarà esaminato uno in particolare nel capitolo successivo. Nelle sezioni successive è definita la sintassi e la semantica ICTL che segue le definizioni proposte in [34] e riportate successivamente in [13]. La definizione dei modelli proposta è utilizzata anche in altri contesti come nella versione intuizionista di LTL [41]. Inoltre, in [13] sono riportati gli enunciati delle proprietà ICTL, le cui dimostrazioni sono state completate durante la stesura dell'elaborato.

3.2 Sintassi ICTL

Per definire la logica ICTL, è essenziale esaminare il contesto in cui essa opera, il quale è delineato dal modello su cui la logica è applicabile. Come menzionato nei capitoli precedenti, questa tipologia di modello è conosciuta come **modello birelazionale**. La combinazione dell'intuizionismo e delle logiche temporali CTL caratterizza ICTL, e l'impiego di tali modelli facilita la comprensione della modellazione delle proprietà temporali riportate in un contesto intuizionista. Ciò permette di delineare le proprietà principali di questa logica, le quali saranno illustrate nel presente capitolo.

Definition 19 (Frame birelazionale). *Un Frame Birelazionale $\mathcal{F}$ è una tripla $\mathcal{F} = \langle W, P, R \rangle$ tale che:*

- *W è un insieme di stati numerabile;*
- *$P \subseteq W \times W$ è un preordine su W, cioè una relazione binaria transitiva e riflessiva;*
- *$R \subseteq W \times W$ è una relazione di transizione binaria e seriale, cioè per ogni $x \in W$ esiste uno stato $y \in W$ tale che xRy;*

In cui le seguenti due condizioni sono soddisfatte per ogni $x, y, z \in W$:

- ***(C1)**: se xRy e yPz allora esiste uno stato $u \in W$ tale che xPu e uRz (vedi in **figura 3.1**)*
- ***(C2)**: se xRy e xPz allora esiste uno stato $u \in W$ tale che zRu e yPu (vedi in **figura 3.2**).*

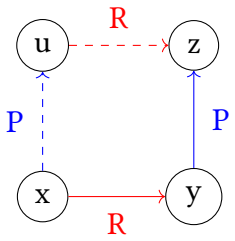


Figure 3.1: Condizione 1.

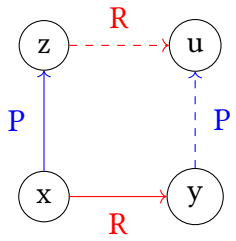


Figure 3.2: Condizione 2.

Definition 20 (Percorso). *Sia $\mathcal{F}$ un frame birelazionale, e w uno stato. Un **percorso** ρ che parte da w è una sequenza infinita di stati di W $\rho = w_0, w_1, w_2, .., w_n, ..$ tale che:*

- $w_0 = w$;
- *per ogni i si ha $\rho_i R \rho_{i+1}$.*

Inoltre:

- *si denota con ρ_i l'i+1-mo stato della sequenza*
- *si denota con $\rho \leq i$ il prefisso finito $\rho_0, \rho_1, ..., \rho_i$ di ρ;*
- *si denota con $\rho \geq i$ il suffisso infinito $\rho_i, rho_{i+1}, ...$ di ρ che parte da ρ_i.*

Lemma 1 (Preordine tra percorsi dal basso). *Sia $\mathcal{F}$ un frame birelazionale e w, w' due stati tali che wPw'. Allora per ogni percorso ρ che parte da w ($\rho_0 = w$) esiste un percorso τ che parte da w' ($\tau_0 = w'$) tale che per ogni $i \in \mathbb{N}$ si ha che $\rho_i P \tau_i$.*

Proof. Il lemma si dimostra per induzione su $\mathbb{N}$.

- caso base: i=0, vero per ipotesi.
- passo induttivo: dobbiamo dimostrare che se il lemma è verificato per un generico $i \in \mathbb{N}$ allora il lemma vale anche per $i + 1$.
  Sia $A = (w_i)_{i \in \mathbb{N}}$ un enumerazione degli stati di W. Per ipotesi induttiva abbiamo che $\rho_i P \tau_i$ e dato che siamo all'interno di un percorso esiste necessariamente un successore di ρ_i tale che $\rho_i R \rho_{i+1}$. Allora per la condizione C_2 riportata in **figura 3.2**, si ha che deve necessariamente esistere uno stato $u \in W$ tale per cui $\tau_i R u$ e $\rho_{i+1} P u$ e dato che ce ne può essere più di uno si sceglie il minimo u dall'enumerazione A definita precedentemente tale per cui si verifica questa condizione. Questo u sarà proprio τ_{i+1}.

□

Ora che sono stati definiti i frame birelazionali è possibile definire i modelli birelazionali che consentiranno di definire la semantica della logica ICTL.

Definition 21 (Modello birelazionale). *Un modello birelazionale (da questo momento in poi verrà chiamato solo modello) è una quadrupla* $\mathcal{M} = \langle W, P, R, \mathcal{V} \rangle$ *tale che* $\langle W, P, R \rangle$ *è un frame birelazionale e* $\mathcal{V}$ *è una funzione di valutazione* $\mathcal{V} : W \longrightarrow 2^{AP}$ *che associa ad ogni stato un sottoinsieme finito di proposizioni atomiche ed è soggetta alla condizione di monotonicità: se* wPw' *allora* $\mathcal{V}(w) \subseteq \mathcal{V}(w')$.

Nell'ambito della logica intuizionista, la tradizionale dualità tra i simboli $\wedge$ e $\vee$, così come tra le quantificazioni esistenziali $\exists$ e universali $\forall$, non mantiene la stessa validità. Pertanto, in ICTL, è essenziale distinguere le formule che coinvolgono quantificazioni universali ed esistenziali, trattandole come entità indipendenti anziché come derivate una dall'altra. Questa considerazione si estende anche all'operatore $\mathcal{R}$, il quale, nella logica temporale classica, rappresenta il duale dell'operatore $\mathcal{U}$.

Definition 22 (Sintassi). *Dato un insieme di proposizioni atomiche* $\mathcal{AP}$ *l'insieme* $\mathcal{FO}$ *delle formule ben formate in ICTL è definito induttivamente come segue:*

- *se* $p \in \mathcal{AP}$ *allora* $p \in \mathcal{FO}$;
- *se* $\varphi, \psi \in \mathcal{FO}$ *allora* $\varphi \wedge \psi \in \mathcal{FO}$;
- *se* $\varphi, \psi \in \mathcal{FO}$ *allora* $\varphi \vee \psi \in \mathcal{FO}$;
- *se* $\varphi, \psi \in \mathcal{FO}$ *allora* $\varphi \rightarrow \psi \in \mathcal{FO}$;
- *se* $\varphi \in \mathcal{FO}$, *allora* $EX\varphi \in \mathcal{FO}$;
- *se* $\varphi \in \mathcal{FO}$, *allora* $AX\varphi \in \mathcal{FO}$;
- *se* $\varphi, \psi \in \mathcal{FO}$ *allora* $E\varphi\mathcal{U}\psi \in \mathcal{FO}$
- *se* $\varphi, \psi \in \mathcal{FO}$ *allora* $A\varphi\mathcal{U}\psi \in \mathcal{FO}$;
- *se* $\varphi, \psi \in \mathcal{FO}$ *allora* $E\varphi\mathcal{R}\psi \in \mathcal{FO}$
- *se* $\varphi, \psi \in \mathcal{FO}$ *allora* $A\varphi\mathcal{R}\psi \in \mathcal{FO}$;

Fanno parte della sintassi ICTL anche le costanti $\bot$ *e* $\top$. *Si definisce la negazione di una formula* φ *come* $\neg\varphi \equiv \varphi \rightarrow \bot$, *equivalenza dimostrata nella* ***proposizione 1***.

3.3 Semantica ICTL

Definition 23 (Soddisfacibilità di una formula ICTL). *sia* $\mathcal{M}$ *un modello birelazionale,* w *uno stato e* φ *una formula. Scriviamo* $\mathcal{M}, s \models \varphi$ *per "*$\mathcal{M}$ *soddisfa* φ *ad s". Tale relazione è definita induttivamente sulla struttura di* φ *come segue:*

- $\mathcal{M}, w \models p \in \mathcal{AP}$ *sse* $p \in \mathcal{V}(w)$;

- $\mathcal{M}, w \not\models \bot$ *per ogni stato* w*;*
- $\mathcal{M}, w \models \varphi \vee \psi$ *sse* $\mathcal{M}, w \models \varphi$ *oppure* $\mathcal{M}, w \models \psi$*;*
- $\mathcal{M}, w \models \varphi \wedge \psi$ *sse* $\mathcal{M}, w \models \varphi$ *e* $\mathcal{M}, w \models \psi$*;*
- $\mathcal{M}, w \models \varphi \rightarrow \psi$ *sse per ogni* w' *tale che* wPw' *si ha che se* $\mathcal{M}, w' \models \varphi$ *allora* $\mathcal{M}, w' \models \psi$*;*
- $\mathcal{M}, w \models EX\varphi$ *sse esiste uno stato* w_1 *tale che* wRw_1 *e* $\mathcal{M}, w_1 \models \varphi$*;*
- $\mathcal{M}, w \models E\varphi\mathcal{U}\psi$ *sse esiste un percorso* ρ *che parte da* w $(\rho_0 = w)$ *tale che* $\mathcal{M}, \rho_i \models \psi$ *per un* $i \geq 0$ *e per ogni* $0 \leq j < i$ $\mathcal{M}, \rho_j \models \varphi$*;*
- $\mathcal{M}, w \models E\varphi\mathcal{R}\psi$ *sse esiste un percorso* ρ *che parte da* s $(\rho_0 = w)$ *tale che per ogni* $i \geq 0$ $(\rho_i \models \psi$ *oppure per un* $0 \leq i \leq j$*,* $\rho_j \models \varphi)$*;*
- $\mathcal{M}, w \models AX\psi$ *sse per ogni stato* w' *tale che* wPw' *e per ogni stato* w_1 *tale che* $w'Rw_1$ *si ha che* $\mathcal{M}, w_1 \models \psi$*;*
- $\mathcal{M}, w \models A\varphi\mathcal{U}\psi$ *sse per ogni stato* w' *tale che* wPw' *e per ogni percorso* ρ *che parte da* w' $(\rho_0 = w')$ $\mathcal{M}, \rho_i \models \psi$ *per un* $i \geq 0$ *e per ogni* $0 \leq j < i$ $\mathcal{M}, \rho_j \models \varphi$*;*
- $\mathcal{M}, w \models A\varphi\mathcal{R}\psi$ *sse per ogni stato* w' *tale che* wPw' *e per ogni percorso* ρ *che parte da* w' $(\rho_0 = w')$*, si ha che per ogni* $i \geq 0$ $(\rho_i \models \psi$ *oppure per un* $0 \leq i \leq j$*,* $\rho_j \models \varphi)$*.*

Si scrive $\mathcal{M}, w \not\models \varphi$ *quando* w *non soddisfa* φ*. Una formula* φ *si dice* ***valida*** *in un modello sse è soddisfatta in ogni* $w \in W$*. Una formula* φ *si dice valida in un frame* $\mathcal{F} = \langle W, P, R \rangle$ *sse è valida in* $\langle W, P, R, \mathcal{V} \rangle$ *per ogni valutazione* $\mathcal{V}$*. Una formula* φ *si dice* ***ICTL-valida*** *sse è valida in ogni frame.*

Observation 1 (Interpretazione dell'operatore $\neg$**).** *Per la definizione della semantica riportata sopra, la negazione di una formula* $\neg\varphi \equiv \varphi \rightarrow \bot$ *è soddisfatta sse per ogni* w' *tale che* wPw' *se* $\mathcal{M}, w' \models \varphi$ *allora* $\mathcal{M}, w' \models \bot$*. Dal momento che nessuno stato soddisfa* $\bot$ *questo equivale a dire "nessuno stato più grande di* w *soddisfa* φ*"*

Equivalentemente a CTL, anche nella versione intuizionista alcuni operatori possono essere derivati dalle formule $\mathcal{FO}$, e la loro equivalenza costituisce delle formule ICTL-valide.

Definition 24 (Operatori derivati). *Definiamo* $\varphi \leftrightarrow \psi$ *come* $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$ *Le seguenti formule sono CTL-valide:*

- $EF\psi \leftrightarrow E\top\mathcal{U}\psi$*;*
- $AF\psi \leftrightarrow A\top\mathcal{U}\psi$*;*
- $EG\psi \leftrightarrow E\bot\mathcal{R}\psi$*;*

- $AG\psi \leftrightarrow A\bot\mathcal{R}\psi$;

Di seguito viene riportata la semantica degli operatori derivati per assicurare la completezza e un maggiore approfondimento dell'elaborato.

Definition 25 (Semantica degli operatori derivati in ICTL). *Dato un modello $\mathcal{M}$ ed uno stato s, la semantica degli operatori derivati è definita come segue:*

- $\mathcal{M}, w \models EF\psi$ *sse esiste un percorso* ρ *che parte da* w $(\rho_0 = w)$ *tale che* $\rho_i \models \psi$ *per un* $i \geq 0$;
- $\mathcal{M}, w \models EG\psi$ *sse esiste un percorso* ρ *che parte da* w $(\rho_0 = w)$ *tale che* $\rho_i \models \psi$ *per ogni* $i \geq 0$;
- $\mathcal{M}, w \models AF\psi$ *sse per ogni stato* w' *tale che* wPw' *e per ogni percorso* ρ *che parte da* w' $(\rho_0 = w')$ $\rho_i \models \psi$ *per un* $i \geq 0$;
- $\mathcal{M}, w \models AG\psi$ *sse per ogni stato* w' *tale che* wPw' *e per ogni percorso* ρ *che parte da* w' $(\rho_0 = w')$ $\rho_i \models \psi$ *per ogni* $i \geq 0$.

3.4 Differenze tra CTL e ICTL

In questa sezione, vengono esplorate le distinzioni tra CTL e ICTL in termini di proprietà. Poiché non è più applicabile la dualità tra alcuni operatori, la validità di determinate formule in CTL non si applica più a ICTL.

Proposition 9 (Formule ICTL non valide). *Le seguenti formule valide in CTL classica non risultano valide in ICTL:*

1. $EX\varphi \leftrightarrow \neg AX\neg\varphi$;
2. $AX\varphi \leftrightarrow \neg EX\neg\varphi$;
3. $E\varphi\mathcal{U}\psi \leftrightarrow \neg A\neg\varphi\mathcal{R}\neg\psi$;
4. $A\varphi\mathcal{U}\psi \leftrightarrow \neg E\neg\varphi\mathcal{R}\neg\psi$;
5. $E\varphi\mathcal{R}\psi \leftrightarrow \neg A\neg\varphi\mathcal{U}\neg\psi$;
6. $A\varphi\mathcal{R}\psi \leftrightarrow \neg E\neg\varphi\mathcal{U}\neg\psi$;
7. $EG\varphi \leftrightarrow \neg AF\neg\varphi$;
8. $AG\varphi \leftrightarrow \neg EF\neg\varphi$;
9. $EF\varphi \leftrightarrow \neg AG\neg\varphi$;
10. $AF\varphi \leftrightarrow \neg EG\neg\varphi$;

Proof. Per dimostrare la non validità di queste formule è sufficiente trovare un **contro-modello**, cioè un modello in cui almeno in un punto le formule non sono soddisfatte. Si consideri il seguente modello:

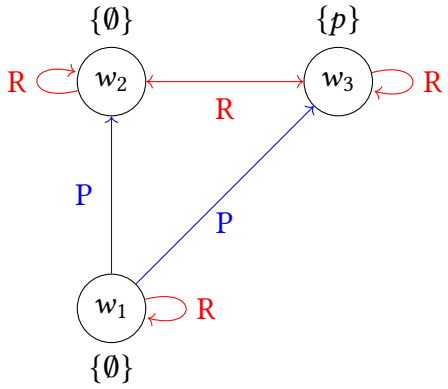


Figure 3.3: Contromodello $\mathcal{M}$

$\mathcal{M}, w_1 \models \neg AXp$ e $\mathcal{M}, w_1 \not\models EX\neg p$ pertanto $\neg AXp \rightarrow EX\neg p$ non è una formula valida in questo modello. Analogamente $\neg A\top \mathcal{U} p \rightarrow E\bot \mathcal{R} \neg p$ non è una formula valida in $\mathcal{M}$. I restanti casi sono del tutto equivalenti. □

Un'altra importante proprietà riguardo la relazione di soddisfacibilità è che questa è monotona rispetto al preordine P. Questa caratteristica è comune a tutte le logiche modali intuizioniste e, intuitivamente, cattura un concetto relativo al fatto che la conoscenza dell'agente non può mai diminuire. Di seguito un'osservazione grafica prima di dimostrare la proprietà:

Observation 2 (Conoscenza dell'agente). *Dati due percorsi* $\rho = \rho_0, \rho_1, \rho_2, \rho_3, ..., \rho_n, ...$ *e* $\tau = \tau_0, \tau_1, \tau_2, \tau_3, ..., \tau_n, ...$ *tali che per ogni* $i \in \mathbb{N}$ *si ha che* $\rho_i P \tau_i$*, che come dimostrato nel* ***lemma 1*** *è verificata l'esistenza. Questi due percorsi, rappresentati in* ***figura 3.4****, illustrano due possibili evoluzioni del sistema. Grazie alla condizione di monotonicità dei modelli birelazionali, per ogni* $i \in \mathbb{N}$ *ciò che è verificato in* ρ_i *lo è anche in* τ_i*. Di conseguenza, tutto ciò che sarà verificato durante l'evoluzione* ρ *del sistema sarà verificato anche durante l'evoluzione* τ *del sistema. Inoltre, potrebbero emergere nuove proprietà verificate in* τ*, non presenti in* ρ*. Quindi,* τ *può essere considerata come un'ulteriore osservazione di una delle possibili evoluzioni del sistema già osservata precedentemente da un agente interagente con esso. Pertanto, con il progredire delle osservazioni, aumentano anche le informazioni che l'agente può acquisire sul sistema.*

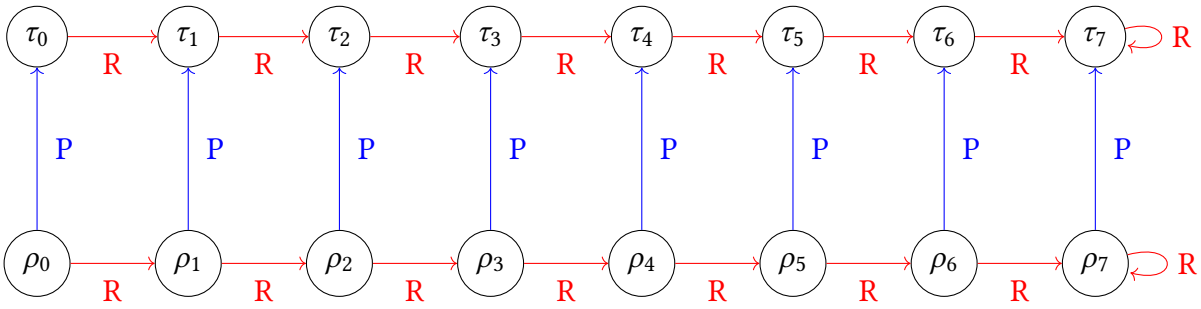


Figure 3.4: preordine tra due percorsi.

Proposition 10 (Monotonia estesa 2). *Sia $\mathcal{M}$ un modello, w, w' due stati e φ una formula ICTL. Allora se $\mathcal{M}, w \models \varphi$ e wPw' Allora $\mathcal{M}, w' \models \varphi$.*

Proof. La proposizione si dimostra per induzione sul numero di operatori logici che compongono la formula.

- Passo base. $p \in AP$ è vero per definizione.
- Passo induttivo. Dobbiamo dimostrare che se la proposizione vale per una formula φ con n operatori, allora vale anche per formule con $n + 1$ operatori.
  - Se φ è una formula nella forma $\neg\psi$, $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\psi_1 \rightarrow \psi_2$, allora è possibile concludere per la **proposizione 2** dimostrata nel capitolo 1.
  - $\varphi \equiv AX\psi$: $\mathcal{M}, w \models AX\psi$ sse per ogni stato w' tale che wPw' e per ogni stato v tale che $w'Rv$ si ha che $\mathcal{M}, v \models \psi$. Consideriamo uno stato w'' tale che $w'Pw''$. Allora per la condizione C_2 dato che $w'Pw''$ e $w'Rv$, vale necessariamente che esiste uno stato v' tale che vPv' e che $w''Rv'$, e dato che in v è soddisfatta ψ, allora lo sarà anche in v' in quanto vale l'ipotesi che la proposizione risulta verificata per ψ. Pertanto $\mathcal{M}, w' \models AX\psi$;
  - $\varphi \equiv A\psi_1\mathcal{U}\psi_2$: Intuitivamente, se si applica lo stesso ragionamento fatto per le formule del tipo $AX\psi$ considerando il **lemma 1** anziché la condizione C_2, risulta chiaro che la proposizione vale anche per formule del tipo $A\psi_1\mathcal{U}\psi_2$ in quanto il lemma risulta una generalizzazione della condizione C_2 applicata lungo un percorso come è stato mostrato anche in **figura 3.4**.
  - $\varphi \equiv A\psi_1\mathcal{R}\psi_2$: la proposizione è dimostrata equivalentemente al caso precedente.
  - $\varphi \equiv EX\psi$: $\mathcal{M}, w \models EX\psi$ sse esiste uno stato w_1 tale che wRw_1 e $\mathcal{M}, w_1 \models \psi$. Dato che wPw', allora per la C_2 esiste uno stato v tale che w_1Pv e $w'Rv$. Dato che la proposizione vale per ψ e dato che w_1Pv allora $\mathcal{M}, v \models \psi$. Allora $\mathcal{M}, w' \models EX\psi$.
  - $\varphi \equiv E\psi_1\mathcal{U}\psi_2$: Intuitivamente, se si applica lo stesso ragionamento fatto per le formule del tipo $EX\psi$ considerando il **lemma 1**, anziché la condizione C_2,

risulta chiaro che la proposizione vale anche per formule del tipo $E\psi_1 \mathcal{U} \psi_2$ in quanto il lemma risulta essere un'estensione della condizione C_2 applicata lungo un percorso come è stato mostrato anche in **figura 3.4**.

– $\varphi \equiv E\psi_1 \mathcal{R} \psi_2$: la proposizione è dimostrata equivalentemente al caso precedente.

□

Nella dimostrazione di alcune proprietà della logica CTL è stata adottata la notazione di linguaggio di una formula. Nella variante intuizionista, seguendo lo stesso approccio, ci si basa su questa definizione per evidenziare specifiche proprietà, integrando tuttavia ulteriori definizioni in conformità il preordine.

Definition 26 (ICTL in termini di linguaggio). *Dato un modello $\mathcal{M}$ e una formula φ si denota l'insieme degli stati di $\mathcal{M}$ che soddisfano φ come $[\![\varphi]\!]^{\mathcal{M}} = \{w \in W | \mathcal{M}, w \models \varphi\}$. se $\mathcal{M}$ è contestualmente dato allora è possibile ometterlo nella definizione e scrivere soltanto $[\![\varphi]\!]$. Dato uno stato $w \in W$ si denota come $w^{\uparrow}$ l'insieme di stati più grandi di w nel rispetto del preordine P, cioè la upward closure di w rispetto a P. Dato un sottoinsieme di stati $Y \subseteq W$ si denota con $Y^{\uparrow}$ l'insieme degli stati di W la cui upward closure è in Y, cioè $Y^{\uparrow} = \{w \in W | w^{\uparrow} \subseteq Y\}$. Infine se Y è un insieme di stati, si denota con Y^c il suo complemento.*

in ICTL le definizioni di pre-immagine sono pressoché identiche alle definizioni fornite per CTL:

Definition 27 (Pre-immagine esistenziale/universale in ICTL). *Sia $\mathcal{M}$ un modello e W un insieme di stati di $\mathcal{M}$. L'insieme $pre_{\exists}(W)$ è detto pre-immagine esistenziale dell'insieme W ed è definito come segue: $pre_{\exists}(W) = \{w \in W | \exists w' \text{ t.c. } R(w, w') \text{ e } w' \in W\}$. L'insieme $pre_{\forall}(W)$ è detto pre-immagine universale dell'insieme W ed è definito come segue: $pre_{\forall}(W) = \{w \in W | \forall w' \, R(w, w') \Rightarrow w' \in W\}$.*

Proposition 11 (Proprietà di ICTL). *Dato un modello $\mathcal{M}$, uno stato w e una formula φ si ha che:*

- $\mathcal{M}, w \models \varphi \rightarrow \psi$ *sse* $w \in ([\![\varphi]\!]^c \cup [\![\psi]\!])^{\uparrow}$;
- $\mathcal{M}, w \models EX\varphi$ *sse* $w \in pre_{\exists}([\![\varphi]\!])$;
- $\mathcal{M}, w \models AX\varphi$ *sse* $w \in pre_{\forall}([\![\varphi]\!])^{\uparrow}$.

Proof. Dato un modello $\mathcal{M}$ e uno stato w:

- (1) ($\longrightarrow$). Per assurdo esiste uno stato $w' \in [\![\varphi]\!] \cap [\![\psi]\!]^c$ e tale che wPw'. Allora $\mathcal{M}, w' \models \varphi$ e $\mathcal{M}, w' \not\models \psi$ e pertanto $\mathcal{M}, w' \not\models \varphi \rightarrow \psi$.
  ($\longleftarrow$). Per ipotesi $w \in ([\![\varphi]\!]^c \cup [\![\psi]\!])^{\uparrow}$. Allora per un generico w' tale che wPw' si ha che $w' \in [\![\varphi]\!]^c$ oppure $w' \in [\![\psi]\!]$. Allora $\mathcal{M}, w' \models \varphi \rightarrow \psi$.

- (2) ($\longrightarrow$). $\mathcal{M}, w \models EX\varphi$ sse esiste uno stato w_1 tale che wRw_1 ed $\mathcal{M}, w_1 \models \varphi$. Allora $w_1 \in [\![\varphi]\!]$ e per definizione $w \in pre_{\exists}(([\![\varphi]\!])$.
($\longleftarrow$). Per definizione $w \in pre_{\exists}(([\![\varphi]\!])$, Allora esiste uno stato w' tale che wRw' e $\mathcal{M}, w' \models \varphi$. Allora $\mathcal{M}, w \models EX\varphi$.

- (3) ($\longrightarrow$). $\mathcal{M}, w \models AX\psi$ sse per ogni stato w' tale che wPw' e per ogni stato w_1 tale che $w'Rw_1$ si ha che $\mathcal{M}, w_1 \models \psi$, cioè tutti i successori di w' nel rispetto di R sono in $[\![\varphi]\!]$. Allora $w' \in pre_{\forall}(([\![\varphi]\!])$ e in particolare $w^{\uparrow} \subseteq pre_{\forall}(([\![\varphi]\!])$. Allora $w \in pre_{\forall}([\![\varphi]\!])^{\uparrow}$.
($\longleftarrow$). $w \in pre_{\forall}([\![\varphi]\!])^{\uparrow}$ sse per ogni w' tale che wPw' e per ogni w_1 tale che $w'Rw_1$ si ha che $w_1 \in [\![\varphi]\!]$. Questa è la definizione per $M, w \models AX\varphi$.

□

Nel capitolo precedente, sono state definite le equivalenze del punto fisso e ne è stata evidenziata l'importanza nella progettazione di un algoritmo di model checking efficiente. Come già illustrato nel contesto di CTL, anche in ICTL si incontrano simili tipologie di equivalenze. Tuttavia, a differenza di CTL, in alcune di tali situazioni, si manifesta un'unica implicazione da un solo lato, mentre in altre, equivalentemente a come accade in CTL, si osserva una completa validità delle formule.

Proposition 12 (Equivalenze del punto fisso in ICTL). *Le seguenti formule sono ICTL-valide:*

- $E\varphi_1 \mathcal{U} \varphi_2 \leftrightarrow \varphi_2 \vee (\varphi_1 \wedge EXE\varphi_1 \mathcal{U} \varphi_2)$;
- $E\varphi_1 \mathcal{R} \varphi_2 \leftrightarrow \varphi_2 \wedge (\varphi_1 \vee EXE\varphi_1 \mathcal{R} \varphi_2)$;
- $\varphi_2 \vee (\varphi_1 \wedge AXA\varphi_1 \mathcal{U} \varphi_2) \rightarrow A\varphi_1 \mathcal{U} \varphi_2$;
- $\varphi_2 \wedge (\varphi_1 \vee AXA\varphi_1 \mathcal{R} \varphi_2) \rightarrow A\varphi_1 \mathcal{R} \varphi_2$.

Proof. Dato un modello $\mathcal{M}$ e uno stato w:

1. ($\longrightarrow$). Per ipotesi $\mathcal{M}, w \models E\varphi_1 \mathcal{U} \varphi_2$ e consideriamo uno stato w' tale che wPw'. Bisogna dimostrare che $\mathcal{M}, w' \models \varphi_2 \vee (\varphi_1 \wedge \text{EXE}\varphi_1 \mathcal{U} \varphi_2)$. Dall'ipotesi si hanno due casi: o $w \in [\![\varphi_2]\!]$, e in questo caso è possibile concludere per la **proposizione 10**, oppure che $w \in [\![\varphi_1]\!]$ ed esiste un percorso ρ che parte da w ($\rho_0 = w$) tale per cui $\rho_i \in [\![\varphi_2]\!]$ per un $i \geq 1$ e $\rho_j \in [\![\varphi_1]\!]$ per ogni $1 \leq j < i$. Da qui si conclude che $w \in [\![\varphi_2 \vee (\varphi_1 \wedge \text{EXE}\varphi_1 \mathcal{U} \varphi_2)]\!]$ e di nuovo, si conclude per la **proposizione 10**.
($\longleftarrow$). Per ipotesi $w \in [\![\varphi_2 \vee (\varphi_1 \wedge \text{EXE}\varphi_1 \mathcal{U} \varphi_2)]\!]$ e consideriamo uno stato w' tale che wPw'. Allora dall'ipotesi $w \in [\![\varphi_2]\!]$ oppure $w \in [\![(\varphi_1 \wedge \text{EXE}\varphi_1 \mathcal{U} \varphi_2)]\!]$, pertanto, anche in questo caso è possibile concludere per la **proposizione 10**.

2. ($\longrightarrow$). Per ipotesi $\mathcal{M}, w \models E\varphi_1 \mathcal{R} \varphi_2$ e consideriamo uno stato w' tale che wPw'. Bisogna dimostrare che $\mathcal{M}, w' \models \varphi_2 \wedge (\varphi_1 \vee \text{EXE}\varphi_1 \mathcal{R} \varphi_2)$, e cioè, per la prorpietà distributiva dell'operatore $\wedge$ rispetto a $\vee$ bisogna dimostrare che $\mathcal{M}, w' \models (\varphi_1 \wedge$

$\varphi_2) \vee (\varphi_2 \wedge \text{EXE}\varphi_1\mathcal{R}\varphi_2)$. Dall'ipotesi si hanno due casi: o $w \in [\![\varphi_1 \wedge \varphi_2]\!]$, e questo significa che $w \in [\![\varphi_1]\!]$ e $w \in [\![\varphi_2]\!]$ e in questo caso è possibile concludere per la **proposizione 10**, oppure che $w \in [\![\varphi_2]\!]$ ed esiste un percorso ρ che parte da w ($\rho_0 = w$) tale per cui $\rho_i \in [\![\varphi_2]\!]$ per ogni $i \geq 1$ oppure $\rho_j \in [\![\varphi_1]\!]$ per un $1 \leq j \leq i$. Da qui si conclude che $w \in [\![\varphi_2 \wedge (\varphi_1 \vee \text{EXE}\varphi_1\mathcal{R}\varphi_2)]\!]$ e di nuovo, si conclude per la **proposizione 10**.
($\longleftarrow$). Per ipotesi $w \in [\![\varphi_2 \wedge (\varphi_1 \vee \text{EXE}\varphi_1\mathcal{R}\varphi_2)]\!]$ e consideriamo uno stato w' tale che wPw'. Allora dall'ipotesi $w \in [\![\varphi_2]\!]$ e $w \in [\![(\varphi_1 \vee \text{EXE}\varphi_1\mathcal{R}\varphi_2)]\!]$, pertanto, anche in questo caso è possibile concludere per la **proposizione 10**.

3. Per ipotesi $\mathcal{M}, w \models \varphi_2 \vee (\varphi_1 \wedge \text{AXA}\varphi_1\mathcal{U}\varphi_2)$ e consideriamo uno stato w' tale che wPw'. Bisogna dimostrare che $w' \in [\![A\varphi_1\mathcal{U}\varphi_2]\!]$. Se $w \in [\![\varphi_2]\!]$ allora per ogni stato w' tale che wPw' e per ogni percorso che si diparte da w', questo soddisferà immediatamente φ_2 e quindi soddisferà $\varphi_1\mathcal{U}\varphi_2$ e pertanto si può concludere usando la **proposizione 10**. Altrimenti se $w \in [\![(\varphi_1 \wedge \text{AXA}\varphi_1\mathcal{U}\varphi_2)]\!]$ significa che $w \in [\![\varphi_1]\!]$ e per ogni stato w' tale che wPw' si ha che se $w'Ru$ allora $u \in [\![A\varphi_1\mathcal{U}\varphi_2]\!]$. Dato che wPw' allora per la **proposizione 10** $w' \in [\![\varphi_1]\!]$ e dato che per ogni stato u tale per cui $w'Ru$ si ha che $u \in [\![A\varphi_1\mathcal{U}\varphi_2]\!]$, allora chiaramente $w' \in [\![A\varphi_1\mathcal{U}\varphi_2]\!]$.

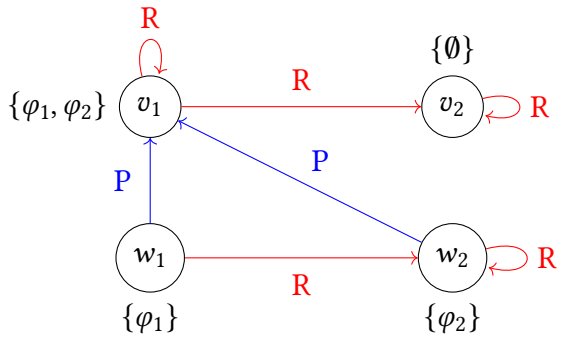
4. Per ipotesi $\mathcal{M}, w \models \varphi_2 \wedge (\varphi_1 \vee \text{AXA}\varphi_1\mathcal{R}\varphi_2)$, cioè per la distributività dell'operatore $\wedge$ rispetto a $\vee$ $\mathcal{M}, w \models (\varphi_1 \wedge \varphi_2) \vee (\varphi_2 \wedge AXA\varphi_1\mathcal{R}\varphi_2)$ e consideriamo uno stato w' tale che wPw'. Bisogna dimostrare che $w' \in [\![A\varphi_1\mathcal{R}\varphi_2]\!]$. Se $w \in [\![\varphi_1 \wedge \varphi_2]\!]$ allora per ogni stato w' tale che wPw' e per ogni percorso che si diparte da w', questo soddisferà immediatamente φ_1 e φ_2 e quindi soddisferà $\varphi_1\mathcal{R}\varphi_2$ e pertanto si può concludere usando la proposizione **10**. Altrimenti se $w \in [\![(\varphi_2 \wedge \text{AXA}\varphi_1\mathcal{R}\varphi_2)]\!]$ significa che $w \in [\![\varphi_2]\!]$ e per ogni stato w' tale che wPw' si ha che se $w'Ru$ allora $u \in [\![A\varphi_1\mathcal{R}\varphi_2]\!]$. Dato che wPw' allora per la **proposizione 10** $w' \in [\![\varphi_2]\!]$ e dato che per ogni stato u tale per cui $w'Ru$ $u \in [\![A\varphi_1\mathcal{R}\varphi_2]\!]$, allora chiaramente $w' \in [\![A\varphi_1\mathcal{R}\varphi_2]\!]$.

□

Proposition 13 (Equivalenze del punto fisso non valide in ICTL). *Le seguenti formule non sono valide in ICTL:*

- $A\varphi_1\mathcal{U}\varphi_2 \rightarrow \varphi_2 \vee (\varphi_1 \wedge AXA\varphi_1\mathcal{U}\varphi_2)$;
- $A\varphi_1\mathcal{R}\varphi_2 \rightarrow \varphi_2 \wedge (\varphi_1 \vee AXA\varphi_1\mathcal{R}\varphi_2)$.

Proof. Per dimostrare la non validità di queste due formule si consideri il seguente modello $\mathcal{M}$:



$M, w_1 \models A\varphi_1 \mathcal{U} \varphi_2$, ma $\mathcal{M}, w_1 \not\models \varphi_2$ né tanto meno $\mathcal{M}, w_1 \not\models \varphi_1 \wedge \text{AXA}\varphi_1 \mathcal{U} \varphi_2$ in quanto dato che $w_1 P v_1$ e dato il percorso $\tau = v_1 \cdot v_2^{\omega}$ si ha che non c'è nessun $i \geq 1$ tale per cui $\mathcal{M}, \tau_i \models \varphi_2$. Similarmente, $\mathcal{M}, w_1 \models A\varphi_1 \mathcal{R} \varphi_2$ ma $\mathcal{M}, w_1 \not\models \varphi_1 \wedge \varphi_2$ né tanto meno $\mathcal{M}, w_1 \not\models \varphi_2 \wedge AXA\varphi_1 \mathcal{R} \varphi_2$. □

3.5 Semantica ICTL sotto una nuova condizione

Nella precedente sezione è stata fornita una panoramica generale circa il funzionamento della logica ICTL. Tuttavia, a differenza di CTL, questa logica presenta due importanti limitazioni riguardanti le formule temporali che quantificano universalmente sui percorsi:

- Le equivalenze del punto fisso non sono più valide da un lato come dimostrato nella **proposizione 13**;
- La relazione di preordine P del modello non permette una verifica "locale" della soddisfacibilità delle formule universali né di definirne una semantica adeguata. In altre parole, partendo da uno stato w, è possibile validare le formule universali solo sui percorsi τ che si dipartono da uno stato w' tali che wPw', e quindi che, nel rispetto del preordine, siano maggiori di ogni percorso ρ che si diparte da w. Tuttavia, non è possibile verificare l'esistenza di un percorso σ che parte da uno stato w'' tale che $w''Pw$ e quindi che, nel rispetto del preordine, sia minore di tutti i percorsi ρ che partono da w. In altre parole, ogni percorso ρ ha come immagine un percorso τ tale per cui per ogni $i \geq 0$ $\rho_i P \tau_i$ ma non si ha alcuna garanzia che ogni percorso τ sia immagine di un percorso ρ.

Tuttavia, è possibile introdurre una condizione aggiuntiva al preordine al fine di superare queste due limitazioni.

Definition 28 (Condizione aggiuntiva al preordine P). *Sia $\mathcal{F} = \langle W, P, R \rangle$ un frame birelazionale, allora la seguente consizione è soddisfatta per ogni $x, y, z \in W$:*

- ***(C3)**: se xPy e yRz allora esiste uno stato $u \in W$ tale che xRu e uPz (vedi in **figura 3.5**)*

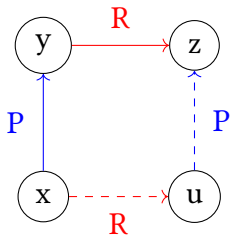


Figure 3.5: Condizione 3.

Lemma 2 (Preordine tra percorsi dall'alto). *Sia $\mathcal{F}$ un frame birelazionale e w, w' due stati tali che wPw'. Allora per ogni percorso τ che parte da w' ($\tau_0 = w'$) esiste un percorso ρ che parte da w ($\rho_0 = w$) tale che per ogni $i \in \mathbb{N}$ si ha che $\rho_i P \tau_i$.*

Proof. Il lemma si dimostra per induzione su $\mathbb{N}$.

- caso base: i=0, vero per ipotesi.
- passo induttivo: dobbiamo dimostrare che se il lemma è verificato per un generico $i \in \mathbb{N}$ allora il lemma vale anche per $i + 1$.
  Sia $A = (w_i)_{i \in \mathbb{N}}$ un enumerazione degli stati di W. Per ipotesi induttiva abbiamo che $\rho_i P \tau_i$ e dato che siamo all'interno di un percorso esiste necessariamente un successore di τ_i tale che $\tau_i R \tau_{i+1}$. Allora per la condizione C_3 riportata in **figura 3.5**, si ha che deve necessariamente esistere uno stato u $\in W$ tale per cui $\rho_i R u$ e $u P \tau_{i+1}$ e dato che ce ne può essere più di uno si sceglie il minimo u dall'enumerazione A definita precedentemente tale per cui si verifica questa condizione. Questo u sarà proprio ρ_{i+1}.

□

Grazie a questo lemma, adesso si ha la garanzia che ogni percorso τ è immagine di un percorso ρ situato più in basso nel rispetto del preordine P. Ora è possibile ridefinire "localmente" la semantica degli operatori universali superando la prima limitazione:

Definition 29 (Soddisfacibilità di una formula $\mathbf{ICTL_{C_3}}$). *sia $\mathcal{M}$ un modello birelazionale che prende in considerazione la condizione C_3 su $\mathcal{F}$, w uno stato e φ una formula. In presenza della condizione C_3 è possibile ridefinire la relazione "$\models$" sulle formule universali e induttivamente sulla struttura di φ come segue:*

- $\mathcal{M}, w \models AX\psi$ *sse per ogni stato* w_1 *tale che* wRw_1 *si ha che* $\mathcal{M}, w_1 \models \psi$;
- $\mathcal{M}, w \models A\varphi \mathcal{U} \psi$ *sse per ogni percorso* ρ *che parte da* w ($\rho_0 = w$) $\mathcal{M}, \rho_i \models \psi$ *per un* $i \geq 0$ *e per ogni* $0 \leq j < i$ $\mathcal{M}, \rho_j \models \varphi$;
- $\mathcal{M}, w \models A\varphi \mathcal{R} \psi$ *sse per ogni percorso* ρ *che parte da* w ($\rho_0 = w$), *si ha che per ogni* $i \geq 0$ ($\rho_i \models \psi$ *oppure per un* $0 \leq i \leq j$, $\rho_j \models \varphi$);

- $\mathcal{M}, w \models AF\psi$ *sse per ogni percorso* ρ *che parte da* w $(\rho_0 = w)$ $\rho_i \models \psi$ *per un* $i \geq 0$*;*
- $\mathcal{M}, w \models AG\psi$ *sse per ogni percorso* ρ *che parte da* w $(\rho_0 = w)$ $\rho_i \models \psi$ *per ogni* $i \geq 0$*.*

In ICTL_{C_3}, la definizione della semantica comporta anche una rivisitazione di alcune proprietà e del concetto di monotonia rispetto al preordine per quanto riguarda le formule universali.

Proposition 14 (Monotonia estesa in ICTL_{C_3}). *Sia* $\mathcal{M}$ *un modello che prende in considerazione la condizione* C_3 *su* $\mathcal{F}$*,* w, w' *due stati e* φ *una formula ICTL. Allora se* $\mathcal{M}, w \models \varphi$ *e* wPw' *Allora* $\mathcal{M}, w' \models \varphi$*.*

Proof. Anche questa proposizione si dimostra per induzione sulla struttura di φ. Il caso base è verificato per definizione. Nel passo induttivo, se una formula φ è nella forma $\neg\psi$, $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\psi_1 \rightarrow \psi_2$, allora è possibile concludere per la **proposizione 2**. Se una formula φ è nella forma $EX\psi$, $E\psi_1\mathcal{U}\psi_2$, $E\psi_1\mathcal{R}\psi_2$, allora è possibile concludere per la **proposizione 10**.

- $\varphi \equiv AX\psi$: $\mathcal{M}, w \models AX\psi$ sse per ogni stato w_1 tale che wRw_1 si ha che $\mathcal{M}, w_1 \models \psi$. Consideriamo uno stato w_2 tale che $w'Rw_2$. Dato che wPw' allora per la condizione C_3 si ha che deve necessariamente esistere uno stato w_1 tale per cui wRw_1 e w_1Pw_2. Questo significa che per ogni percorso τ che parte da w' $(\tau_0 = w')$ e tale che $\tau_1 = w_2$ esiste un percorso ρ che parte da w $(\rho_0 = w)$ tale che $\rho_1 = w_1$ e w_1Pw_2. Pertanto dato che $\mathcal{M}, w_1 \models \psi$ e dato che per l'ipotesi induttiva la proposizione risulta verificata per ψ allora si ottiene che $\mathcal{M}, w_2 \models \psi$. Pertanto si conclude che $\mathcal{M}, w' \models AX\psi$.
- $\varphi \equiv A\psi_1\mathcal{U}\psi_2$: Intuitivamente, se si applica lo stesso ragionamento fatto per le formule del tipo $AX\psi$ considerando il **lemma 2** anziché la condizione C_3, risulta chiaro che la proposizione vale anche per formule del tipo $A\psi_1\mathcal{U}\psi_2$ in quanto il lemma risulta una generalizzazione della condizione C_3 applicata lungo un percorso come è stato mostrato anche in **figura 3.4**.
- $\varphi \equiv A\psi_1\mathcal{R}\psi_2$: la proposizione è dimostrata equivalentemente al caso precedente.

□

Una volta stabilita la semantica di ICTL con la condizione C_3, è necessario dimostrare l'equivalenza tra le definizioni di soddisfacibilità di ICTL con e senza questa condizione. Poiché la definizione di soddisfacibilità è equivalente per tutti gli operatori tranne che per quelli temporali universali, bisogna dimostrare che questa equivalenza si mantiene anche per questi ultimi.

Theorem 4 (Equivalenza delle definizioni di soddisfacibilità di ICTL e ICTL_{C_3}). *Sia* $\mathcal{M}$ *un modello di ICTL,* w *uno stato di* $\mathcal{M}$*, e* $A\psi$ *una formula di ICTL dove* ψ *è una formula temporale del tipo* $X\varphi$*,* $\varphi_1\mathcal{U}\varphi_2$*,* $\varphi_1\mathcal{R}\varphi_2$*. Allora, la seguente affermazione è vera:* $\mathcal{M}, w \models A\psi$ *sse* $\mathcal{M}_{C_3}, w \models A\psi$*, dove* $\mathcal{M}_{C_3}$ *denota il modello* $\mathcal{M}$ *sottoposto alla condizione* C_3*.*

Proof. Dimostriamo entrambi i lati:

- ($\rightarrow$)
  $\mathcal{M}, w \models A\psi$, allora dato che la soddisfacibilità locale per le due definizioni è identica, banalmente si ha che $\mathcal{M}_{C_3}, w \models A\psi$.

- ($\leftarrow$)
  $\mathcal{M}, w \models A\psi$ sse per ogni stato w' tale che wPw' e per ogni percorso ρ che parte da $w'(\rho_0 = w')$ si ha che $\mathcal{M}, \rho_0 \models \psi$. per ipotesi $\mathcal{M}_{C_3}, w \models A\psi$. Allora per la **proposizione 14** qualunque stato w' tale che wPw' soddisferà $A\psi$ e pertanto possiamo concludere.

□

Proposition 15 (Formule ICTL$_{C_3}$ non valide). *La **proposizione 9** dimostrata per ICTL risulta valere anche in ICTL$_{C_3}$.*

Proof. Il contromodello $\mathcal{M}$ in **figura 3.3** soddisfa anche la condizione C_3, e dato che attraverso il **teorema 4** è stata dimostrata l'equivalenza tra le due definizioni di soddisfacibilità tra le due logiche è possibile concludere che le formule che non risultano valide in **proposizione 9** per ICTL non risultano valide neanche per $ICTL_{C_3}$ □

Proposition 16 (Proprietà di ICTL$_{C_3}$). *Dato un modello $\mathcal{M}$, uno stato w e una formula φ si ha che $\mathcal{M}, w \models AX\varphi$ sse $w \in pre_\forall([\![\varphi]\!])$.*

Proof. ($\longrightarrow$). $\mathcal{M}, w \models AX\psi$ sse per ogni stato w_1 tale che wRw_1 si ha che $\mathcal{M}, w_1 \models \psi$, cioè tutti i successori di w nel rispetto di R sono in $[\![\varphi]\!]$. Allora $w \in pre_\forall(([\![\varphi]\!])$ per definizione. ($\longleftarrow$). $w \in pre_\forall(([\![\varphi]\!])$ sse per ogni w_1 tale che wRw_1 si ha che $w_1 \in [\![\varphi]\!]$. Questa è la definizione per $M, w \models AX\varphi$. □

A differenza di ICTL infine, è possibile dimostrare che la semantica di ICTL $_{C_3}$, porta alla validità delle equivalenze del punto fisso che in ICTL non risultavano valide come dimostrato nella **proposizione 13**.

Proposition 17 (Equivalenze del punto fisso in ICTL$_{C_3}$). *Le seguenti formule ICTL-valide:*

1. $A\varphi_1 \mathcal{U} \varphi_2 \rightarrow \varphi_2 \vee (\varphi_1 \wedge AXA\varphi_1 \mathcal{U} \varphi_2)$;
2. $A\varphi_1 \mathcal{R} \varphi_2 \rightarrow \varphi_2 \wedge (\varphi_1 \vee AXA\varphi_1 \mathcal{R} \varphi_2)$.

Proof. Dato un modello $\mathcal{M}$ e uno stato w:

1. Per ipotesi $\mathcal{M}, w \models A\varphi_1 \mathcal{U} \varphi_2$ e consideriamo uno stato w' tale che wPw'. Bisogna dimostrare che $\mathcal{M}, w' \models \varphi_2 \vee (\varphi_1 \wedge \text{AXA}\varphi_1 \mathcal{U} \varphi_2)$. Dall'ipotesi si hanno due casi: o $w \in [\![\varphi_2]\!]$, e in questo caso è possibile concludere per la **proposizione 14**, oppure che $w \in [\![\varphi_1]\!]$ e per ogni percorso ρ che parte da w ($\rho_0 = w$) tale per cui $\rho_i \in [\![\varphi_2]\!]$ per un $i \geq 1$ e $\rho_j \in [\![\varphi_1]\!]$ per ogni $1 \leq j < i$. Da qui si conclude che $w \in [\![\varphi_2 \vee (\varphi_1 \wedge \text{AXA}\varphi_1 \mathcal{U} \varphi_2)]\!]$ e di nuovo, si conclude per la **proposizione 14**.

2. Per ipotesi $\mathcal{M}, w \models E\varphi_1 \mathcal{R} \varphi_2$ e consideriamo uno stato w' tale che wPw'. Bisogna dimostrare che $\mathcal{M}, w' \models \varphi_2 \wedge (\varphi_1 \vee \text{AXA}\varphi_1 \mathcal{R} \varphi_2)$, e cioè, per la prorpietà distributiva dell'operatore $\wedge$ rispetto a $\vee$ bisogna dimostrare che $\mathcal{M}, w' \models (\varphi_1 \wedge \varphi_2) \vee (\varphi_2 \wedge \text{AXA}\varphi_1 \mathcal{R} \varphi_2)$. Dall'ipotesi si hanno due casi: o $w \in [\![\varphi_1 \wedge \varphi_2]\!]$, e questo significa che $w \in [\![\varphi_1]\!]$ e $w \in [\![\varphi_2]\!]$ e in questo caso è possibile concludere per la **proposizione 14**, oppure che $w \in [\![\varphi_2]\!]$ e per ogni percorso ρ che parte da w ($\rho_0 = w$) tale per cui $\rho_i \in [\![\varphi_2]\!]$ per ogni $i \geq 1$ oppure $\rho_j \in [\![\varphi_1]\!]$ per un $1 \leq j \leq i$. Da qui si conclude che $w \in [\![\varphi_2 \wedge (\varphi_1 \vee \text{AXA}\varphi_1 \mathcal{R} \varphi_2)]\!]$ e di nuovo, si conclude per la **proposizione 14**.

□

3.6 ICTL-Model Checking

Questa sezione presenta una versione efficiente del model checking per ICTL. Come avviene per CTL, l'approccio consiste nell'analizzare su ciascuno stato quali formule sono vere. Poiché la soddisfacibilità di una formula in ICTL è definita induttivamente, essa dipende a sua volta dalla soddisfacibilità delle sue sottoformule.

Definition 30 (Sottoformule di φ in ICTL). *Sia φ una formula. L'insieme delle sottoformule di φ che si denota con $SF(\varphi)$ è definito induttivamente come segue:*

- $\varphi \in SF(\varphi)$;
- *se* $\psi_1 \wedge \psi_2 \in SF(\varphi)$ *allora* $\psi_1, \psi_2 \in SF(\varphi)$;
- *se* $\psi_1 \vee \psi_2 \in SF(\varphi)$ *allora* $\psi_1, \psi_2 \in SF(\varphi)$;
- *se* $\psi_1 \rightarrow \psi_2 \in SF(\varphi)$ *allora* $\psi_1, \psi_2 \in SF(\varphi)$;
- *se* $EX\psi \in SF(\varphi)$ *allora* $\psi \in SF(\varphi)$;
- *se* $AX\psi \in SF(\varphi)$ *allora* $\psi \in SF(\varphi)$;
- *se* $E\psi_1 \mathcal{U} \psi_2 \in SF(\varphi)$ *allora* $\psi_1, \psi_2 \in SF(\varphi)$;
- *se* $A\psi_1 \mathcal{U} \psi_2 \in SF(\varphi)$ *allora* $\psi_1, \psi_2 \in SF(\varphi)$;
- *se* $E\psi_1 \mathcal{R} \psi_2 \in SF(\varphi)$ *allora* $\psi_1, \psi_2 \in SF(\varphi)$;
- *se* $A\psi_1 \mathcal{R} \psi_2 \in SF(\varphi)$ *allora* $\psi_1, \psi_2 \in SF(\varphi)$.

Anche in questo caso, dalla definizione risulta che il numero di sottoformule di una formula φ è lineare nella lunghezza di φ: $|SF(\varphi)| = \theta(|\varphi|)$.
In modo simile al CTL model checking, il model checking per ICTL combina l'idea di calcolare una formula a partire dalle sue sottoformule con il calcolo del punto fisso. Ciò significa che la verifica delle proprietà temporali al "passo successivo" viene eseguita calcolando l'appropriata pre-immagine, mentre la verifica delle proprietà temporali "a lungo termine" viene effettuata calcolandone il corrispondente punto fisso. A differenza del model checking per CTL, i casi delle formule che quantificano universalmente sui percorsi non sono più considerati opzionali. Inoltre, dato che la negazione di una formula $\neg\varphi$ può essere definita in termini di $\varphi \rightarrow \bot$, è necessario gestire adeguatamente anche questo caso. Infine, bisogna affrontare in modo appropriato i casi in cui φ è una formula del tipo $E\varphi_1\mathcal{R}\varphi_2$ o del tipo $A\varphi_1\mathcal{R}\varphi_2$, sostituendo il classico caso degli operatori $EG\psi$ e $AG\psi$.
L'algoritmo riceve in input un modello $\mathcal{M}$ e una formula φ. Egli Procede ricorsivamente sulle sottoformule di φ, trattando ogni tipo di sottoformula in modo specifico. L'algoritmo termina restituendo tutti gli stati che soddisfano φ. Se nessuno stato soddisfa la formula, viene restituito l'insieme vuoto $\emptyset$. L'algoritmo è riportato di seguito:

Algorithm 2 mCheckICTL

Require: $\mathcal{M}, \varphi$
Ensure: $\{s|\ \mathcal{M}, s \models \varphi\}$

switch φ **do**
  case $\varphi \equiv p$ return $\{s|p \in \mathcal{V}(s)\}$;
  case $\varphi \equiv \psi_1 \vee \psi_2$: return $mCheckICTL(\mathcal{M}, \psi_1) \cup mCheckICTL(\mathcal{M}, \psi_2)$;
  case $\varphi \equiv \psi_1 \wedge \psi_2$: return $mCheckICTL(\mathcal{M}, \psi_1) \cap mCheckICTL(\mathcal{M}, \psi_2)$;
  case $\varphi \equiv \psi_1 \rightarrow \psi_2$:
    $return((S \setminus mCheckICTL(\mathcal{M}, \psi_1)) \cup mCheckICTL(\mathcal{M}, \psi_2))^{\uparrow}$;
  case $\varphi \equiv E\mathcal{X}\psi$: return $pre_{\exists}(mCheckCTL(\mathcal{M}, \psi))$;
  case $\varphi \equiv A\mathcal{X}\psi$: return $pre_{\forall}(mCheckCTL(\mathcal{M}, \psi))$;
  case $\varphi \equiv E\psi_1\mathcal{U}\psi_2$:
    $Q_1 := \emptyset$;
    $Q_2 := mCheckCTL(\mathcal{M}, \psi_1)$; $Q_3 := mCheckCTL(\mathcal{M}, \psi_2)$;
    while $Q_3 \not\subseteq Q_1$ **do**
      $Q_1 := Q_1 \cup Q_3$;
      $Q_3 := pre_{\exists}(Q_1) \cap (Q_2)$;
    end while
    return Q_1;
  case $\varphi \equiv A\psi_1\mathcal{U}\psi_2$:
    $Q_1 := \emptyset$;
    $Q_2 := mCheckCTL(\mathcal{M}, \psi_1)$; $Q_3 := mCheckCTL(\mathcal{M}, \psi_2)$;
    while $Q_3 \not\subseteq Q_1$ **do**
      $Q_1 := Q_1 \cup Q_3$;
      $Q_3 := pre_{\forall}(Q_1) \cap (Q_2)$;
    end while
    return Q_1;
  case $\varphi \equiv E\psi_1\mathcal{R}\psi_2$:
    $Q_1 := S$;
    $Q_2 := mCheckCTL(\mathcal{M}, \psi_1)$; $Q_3 := mCheckCTL(\mathcal{M}, \psi_2)$;
    while $Q_1 \not\subseteq Q_3$ **do**
      $Q_1 := Q_1 \cap Q_3$;
      $Q_3 := pre_{\exists}(Q_1) \cup (Q_2)$;
    end while
    return Q_1;
  case $\varphi \equiv A\psi_1\mathcal{R}\psi_2$:
    $Q_1 := S$;
    $Q_2 := mCheckCTL(\mathcal{M}, \psi_1)$; $Q_3 := mCheckCTL(\mathcal{M}, \psi_2)$;
    while $Q_1 \not\subseteq Q_3$ **do**
      $Q_1 := Q_1 \cap Q_3$;
      $Q_3 := pre_{\forall}(Q_1) \cup (Q_2)$;
    end while
    return Q_1;

Theorem 5 (Classe di complessità dell'ICTL model checking). *L'algoritmo di ICTL model checking è **PTIME-COMPLETE**.*

Proof. Bisogna provare:

- **Hardness**: sia K un modello di Kripke per CTL. Allora è possibile ridefinirlo come un modello birelazionale definendo il preordine P come una relazione binaria che corrisponde alla funzione identità. Cioè dati due stati s ed s' diremo che sPs' sse $s = s'$. Questa relazione è riflessiva e transitiva, per cui corrisponde alla definizione di preordine in ICTL e soddisfa banalmente le condizioni C_1, C_2 e C_3. Pertanto è possibile concludere che il problema di model checking CTL è riducibile ad un problema di ICTL model checking, pertanto la hardness è verificata.

- **Completeness**: per verificare che l'algoritmo si trova effettivamente in PTIME è necessario fare un'analisi dell'algoritmo. L'algoritmo effettua una chiamata ricorsiva per ogni sottoformula di φ per un totale di $|SF(\varphi)|$ chiamate. Dobbiamo dimostrare che ogni chiamata ricorsiva impiega al più tempo $O(|R|)$. Nel caso di una proposizione atomica e dei connettivi booleani è abbastanza chiaro che sia così. Anche il calcolo di $Q^{\uparrow}$ richiede al più tempo polinomiale, in particolare richiede al più tempo $O(|S|^2)$. Per quanto riguarda il calcolo delle pre-immagini (sia esistenziale che universale) per un dato insieme Q può essere fatto al più in tempo $O(|R|)$ e infine, il while richiede al più $O(|S|)$ iterazioni. In generale il tempo impiegato dal while è al più $O(|R|)$ in quanto il calcolo delle pre immagini può essere implementato in modo tale che ogni transizione del modello venga visitata una e una sola volta durante le varie iterazioni del ciclo. Pertanto l'algoritmo si trova in **PTIME**.

□

Corollary 2 (Complessità computazionale dell'ICTL model checking). *L'algoritmo di model checking ICTL può essere risolto in tempo $O(|\varphi| * |R|)$ quindi in tempo polinomiale sull'input.*

–4–
Implementazione & Benchmark

In questo capitolo viene esplorato il legame stretto tra il concetto di informazione e la rappresentazione della conoscenza con l'intuizionismo. Piuttosto che offrire un'alternativa alle logiche epistemiche o modali, l'intuizionismo introduce un nuovo approccio per comprendere e descrivere l'informazione implicita intrinseca a un modello. Successivamente, dopo aver chiarito e motivato questa prospettiva, verranno presentati diversi scenari in cui ICTL può essere utilizzata per rappresentare questo tipo di conoscenza, integrandola con la dinamica temporale di un modello. In seguito, verrà mostrata un'implementazione software del model checking ICTL, evidenziando le eccellenti performance ottenute nella generazione e nella verifica di modelli birelazionali di grandi dimensioni.

4.1 Informazione intuizionista

La nozione di informazione [12] non è un tema standard nella logica, ma è possibile trovarla ovunque. Un esempio significativo è l'intreccio tra informazione "*fattuale*" e "*procedurale*", così come tra informazione "*statica*" e "*dinamica*". La logica epistemica [42], ad esempio, descrive l'informazione degli agenti in termini di insiemi di mondi possibili e i processi dinamici [43] che aggiornano questi insiemi, come l'osservazione e la conversazione. La teoria delle situazioni [44], invece, si concentra sulle correlazioni tra stati locali del mondo e su come gli agenti si mantengono sincronizzati con essi. Queste visioni si fondono nei cosiddetti "*modelli di eventi temporali*", ma non esauriscono il panorama. Un'altra prospettiva cruciale è la nozione di informazione nella prova-teorica o algoritmica, che concepisce l'informazione come un codice manipolato da processi dinamici di elucidazione, come la dimostrazione matematica o il calcolo computazionale. In questo contesto, l'informazione è vista come una sequenza di simboli o istruzioni che possono essere trasformate e riorganizzate attraverso regole formali e procedure algoritmiche. Questi processi di elucidazione, non producono informazione nello stesso modo in cui avviene con l'aggiornamento semantico: l'aggiornamento semantico riguarda il cambiamento delle conoscenze di un agente riguardo a possibili stati del mondo, ad esempio attraverso l'osservazione o la comunicazione. In altre parole, l'aggiornamento seman-

tico modifica ciò che un agente considera possibile o vero. Al contrario, i processi di elucidazione lavorano su dati strutturati più finemente, e questi sono rappresentati in linguaggi formali o naturali e possono includere formule matematiche, algoritmi, o altre forme di rappresentazione simbolica. Questi processi trasformano questi dati secondo regole formali, ma non cambiano direttamente la conoscenza di un agente riguardo ai possibili stati del mondo come avviene con l'aggiornamento semantico, ma forniscono una comprensione più approfondita e dettagliata delle strutture informative esistenti.
La relazione tra i resoconti semantici (che si occupano dell'aggiornamento delle conoscenze basate su possibili stati del mondo) e quelli prova-teorici (che si occupano della manipolazione formale di dati strutturati) rappresenta una sfida significativa. Integrare questi due approcci è complesso, ma ci sono tentativi promettenti in cui si cerca di unificare le diverse visioni dell'informazione in un quadro coerente.
Questa diversità riflette le molte posizioni complementari nella scienza in generale, dalla logica alla teoria dell'informazione di **Shannon** e alla complessità di **Kolmogorov**, la cui integrazione solleva importanti questioni fondamentali per la filosofia dell'informazione.
Una delle prime e più influenti teorie logiche basate sull'informazione è la logica intuizionista. Questa prima parte del capitolo esaminerà la visione dell'informazione sottesa all'intuizionismo e le questioni che essa solleva, mettendo in luce come questa prospettiva si relazioni con la logica epistemica. Saranno analizzate le connessioni tra questi modelli, e discusse le implicazioni generali di queste interrelazioni per la teoria dell'informazione.
Come è stato già esposto nel capitolo 1, l'intuizionismo vede la matematica come una rete di schemi di dimostrazioni costruttive e definizioni corrispondenti degli oggetti. In questo modo, infonde alle costanti logiche proposizionali o del primo ordine uno spirito di "prova teoretica". Ad esempio, una dimostrazione di una congiunzione $A \wedge B$ è una coppia di dimostrazioni rispettivamente per A e per B, e una dimostrazione di un'implicazione $A \longrightarrow B$ è una procedura per trasformare le dimostrazioni di A in dimostrazioni di B.
Inoltre i modelli intuizionisti, spiegano informalmente, come modo per motivare il formalismo matematico, che un modello descrive un processo di indagine da parte di un agente e che i suoi mondi sono "situazioni informative" in cui è stata già raggiunta una certa "conoscenza". Intuitivamente, una struttura di Kripke intuizionista o anche un modello birelazionale nel rispetto del preordine P descrive un processo informativo in cui un agente apprende progressivamente lo stato del mondo reale, codificato in una valutazione proposizionale. Ai punti finali della struttura[1], tutte le informazioni sono raccolte e l'agente conosce tutti i fatti sul mondo reale. Una ragione per cui la storia rimane esile è che, nella logica intuizionista come in molte altre aree della logica, la semantica ha seguito la teoria delle dimostrazioni e i corrispondenti modelli sono stati ideati per garantire che un linguaggio già dato abbia un significato e che un calcolo delle dimostrazioni dato si dimostri completo.
Tradizionalmente, nella logica intuizionista e in altre aree della logica, la semantica è stata sviluppata seguendo la teoria delle dimostrazioni. Ciò significa che i modelli semantici

[1]Questi punti banalmente esistono perché il modello è finito. Quindi si arriverà ad uno stato s che non avrà stati "maggiori" nel rispetto del preordine, se non sé stesso.

sono stati concepiti per dare significato a un linguaggio già dato[2] e per dimostrare la completezza di un certo calcolo delle dimostrazioni. In questa prospettiva, i modelli semantici sono strumenti al servizio della pratica dimostrativa, e il termine "semantica" stesso riflette questo orientamento verso l'utilità pratica.
Tuttavia, si propone anche un punto di vista alternativo, secondo il quale le strutture semantiche stesse dovrebbero essere considerate come oggetti di studio primari. Questo significa che anziché sviluppare la semantica per adattarsi a un linguaggio preesistente o a una pratica dimostrativa, ci si concentra sulla comprensione e l'analisi delle strutture semantiche per loro stesse. In altre parole, invece di vedere la semantica come uno strumento per dare significato a un linguaggio, si considera la semantica come una disciplina indipendente che può guidare lo sviluppo di linguaggi e pratiche dimostrative.
Questo approccio filosofico discusso anche in [45] e in [46] sottolinea l'importanza di esaminare criticamente i presupposti e le pratiche linguistiche e dimostrative, e suggerisce che il linguaggio e la semantica dovrebbero essere adattati in base alle esigenze concettuali e alla natura dei fenomeni studiati. In particolare, riguardo alla logica intuizionista, si suggerisce che il linguaggio intuizionista dovrebbe essere concepito come il miglior sistema di "interruttori logici" per il ragionamento costruttivo e l'analisi dell'informazione, anziché come una semplice alternativa ai linguaggi classici della logica.
Si possono esplorare diversi concetti fondamentali relativi alla relazione tra l'informazione e la sua rappresentazione nella logica intuizionista:

- **Equivalenza tra processi informativi**: quando due processi informativi sono considerati equivalenti? Questo può essere determinato considerando solamente se l'insieme delle valutazioni proposizionali sulle foglie (stati finali) è lo stesso, oppure in senso più stretto, considerando anche se i due processi sono simili anche nella disposizione delle fasi intermedie? Come discusso precedentemente nel capitolo 1, emerge che la bisimulazione è un'equivalenza standard della teoria dei processi, che risulta fondamentale nel trovare una risposta al quesito. Tuttavia esistono altre forme di equivalenza tra processi. Pertanto, questa è una questione fondamentale e legittima, sebbene spesso trascurata, che richiede una discussione esplicita su quale concetto di equivalenza di processo sia più appropriato per la raccolta di informazioni nel tempo;

- **Distinzione tra informazioni fattuali e procedurali**: bisogna sottolineare una distinzione fondamentale nell'informazione rappresentata nella logica intuizionista. Questa distinzione si articola in due tipi di informazione:

    1. **Informazioni fattuali**: Riguardano la conoscenza sullo stato del mondo, cioè su "come le cose sono";
    2. **Informazioni procedurali**: Riguardano il processo attraverso il quale si acquisisce la conoscenza, cioè "come si arriva a conoscere i fatti".

[2] Infatti la sintassi della IPL è identica al frammento proposizionale classico

Quanto la distinzione tra informazioni fattuali e procedurali è effettivamente robusta come caratteristica dell'informazione? Questo potrebbe dipendere dal contesto ma comunque il quesito resta illuminante per comprendere il processo di acquisizione della conoscenza;

- **Vincoli sui processi informativi**: Si nota che la logica intuizionista pone pochi vincoli sui processi informativi, consentendo una varietà di approcci e risultati. Tuttavia, è possibile imporre restrizioni maggiori sui processi, ad esempio limitando i passaggi ammissibili a quelli che corrispondono a conoscere solo una proposizione alla volta, ottenendo così logiche più forti che descrivono specifici tipi di processo;
- **Implicazioni delle logiche intermedie**: Si suggerisce che queste considerazioni possono portare a una nuova visione delle logiche intermedie tra logica intuizionista e logica classica. Queste logiche intermedie potrebbero emergere come teorie dell'informazione ben motivate che descrivono particolari tipi di processo.

Come detto prima, fornire modelli semantici per un linguaggio non implica che questo sia il migliore per quei modelli. Quale linguaggio descrive meglio i processi informativi di cui l'intuizionismo si occupa o dovrebbe occuparsi? La logica proposizionale intuizionista si concentra esclusivamente sulla conoscenza stabile raggiunta in una certa fase, indipendentemente da come il processo continua da lì in poi. E lo fa in un modo piuttosto peculiare, cioè "caricando" il resoconto delle costanti logiche. Così, $\neg A$ non dice solo che non è vera A nella fase attuale, ma anche che non sarà mai vera, dove "mai" si riferisce al futuro del processo attuale[3]. **Van Benthem** [47] chiama questa caratteristica dell'intuizionismo di "caricamento dell'informazione" **conoscenza implicita**, e la contrappone alla conoscenza esplicita trovata nella logica epistemica, che mantiene il resoconto classico delle operazioni booleane, ma aggiunge un operatore modale esplicito $K_i\varphi$ che dice che "l'agente i sa che φ è vera", o semanticamente, $\mathcal{M}, s \models K_i\varphi$ sse se φ è vera in tutti i mondi t epistemicamente accessibili da s in $\mathcal{M}$. Ovviamente, il caricamento permette al linguaggio intuizionista di esprimere nozioni interessanti. Ad esempio, la formula $\neg\neg\varphi$ dice che ogni fase ha una fase successiva in cui φ è vera, o negli alberi finiti, ogni nodo finale rende φ vera. Questo è vicino a dire che, a meno di dettagli della struttura del processo, "sappiamo" già ora che φ deve essere realmente vera. L'intuizionismo distingue questo dall'avere verificato φ in sé. È possibile anche mettere in relazione la logica intuizionista con la logica modale, evidenziando come le formule intuizioniste descrivano i processi informativi in modo indiretto attraverso il "caricamento" delle costanti logiche classiche. **Van Benthem** propose un'alternativa più diretta usando la logica modale [48], [49], che descrive i processi informativi in modo più semplice e chiaro, tuttavia ci sono alcuni vantaggi nell'utilizzare la logica intuizionista rispetto alla logica modale per descrivere i processi informativi:

- La logica intuizionista distingue tra avere verificato φ e il dire semplicemente che φ sarà vera in futuro, una distinzione importante per descrivere processi informativi;

[3] Gli stati maggiori dello stato attuale nel rispetto del preordine

- Sebbene la logica modale fornisca una descrizione più diretta dei processi informativi, la logica intuizionista ne costituisce un frammento persistente più espressivo. Le formule modali esistenziali $\Diamond$ dicono che φ potrebbe ancora diventare vera in futuro, ma queste formule non sono persistenti come le asserzioni intuizioniste.

- La logica intuizionista ha origini e motivazioni diverse dalla logica modale, essendo nata come teoria del ragionamento matematico costruttivo. Quindi può catturare aspetti di questo tipo di ragionamento che la logica modale non coglie.

In sintesi, sebbene la logica modale fornisca una descrizione più diretta dei processi informativi, la logica intuizionista ha il vantaggio di esprimere nozioni specifiche su questi processi, distinguere tra diversi tipi di conoscenza, in quanto si concentra sulla nozione di "conoscenza stabile" raggiunta in un certo stadio del processo informativo e avere legami con il ragionamento matematico costruttivo. Quindi entrambe le prospettive sono utili e complementari per studiare l'informazione.
La logica intuizionista può essere vista come un linguaggio che rende esplicito un tipo di conoscenza implicita, consentendo così di definire proprietà nei modelli che catturano questa dimensione della conoscenza. Nei successivi paragrafi del capitolo, verranno esposti scenari e casi d'uso interessanti che illustrano questo concetto attraverso l'utilizzo di ICTL. Verrà mostrato che, oltre a descrivere la dinamica temporale di un modello come in CTL, è anche possibile descrivere e modellare esplicitamente il guadagno di informazione implicita. Questo "guadagno" corrisponde effettivamente alle osservazioni osservazione del sistema.
Intuitivamente l'idea è che, se una proprietà è verificata sin dall'inizio della computazione, lo sarà sempre durante tutte le osservazioni del modello. Inoltre, se questa proprietà è modellabile, allora esiste un modello che permette di verificarla a partire da un certo punto in poi, mantenendola persistente. Ciò significa che, se la proprietà si verifica in uno stato e viene confermata su ogni percorso che si diparte da quello stato, continuerà a essere verificata in ogni osservazione futura. In parole povere stiamo praticamente modellando dei sistemi in cui la presenza di un'informazione garantisce la persistenza. Se una proprietà è modellata in questo modo, possiamo garantirne la sicurezza in termini di persistenza indipendentemente dall'evoluzione del sistema.

4.2 Casi d'uso

- **Monitoraggio periodico di un paziente per accertamenti oncologici**. Consideriamo un modello $\mathcal{M}$ in cui un medico ha effettuato n visite oncologiche ad un paziente. Ciascuna visita medica prevede $m + 1$ esami oncologici e per semplicità consideriamo che tutte le visite mediche richiedano gli stessi esami ogni volta. Una visita medica è modellata attraverso 3 stati. Pertanto gli stati del modello sono di 3 tipologie:

    1. gli stati in cui la visita medica è iniziata;

2. gli stati in cui il medico sta effettuando gli esami di accertamento sul paziente;
3. gli stati in cui gli esami medici sono conclusi e andati a buon fine.

Pertanto, gli stati s_j sono della tipologia 1, gli stati s_{j+1} sono della tipologia 2 e gli stati s_{j+2} sono della tipologia 3 con j multiplo di 3. Consideriamo le seguenti proposizioni atomiche:

1. h_i sta per "la visita oncologica i-ma è andato a buon fine";
2. e_i sta per "il medico ha effettuato l'esame di accertamento i-mo sul paziente".
3. c sta per "sono state rilevate delle masse tumorali sul paziente".

La dinamica temporale del modello di una generica visita i è descritta come segue:

- Il medico inizia la visita sul paziente, effettuando i vari esami di accertamento, questo nel modello si traduce come una transizione R dallo stato s_j allo stato allo stato s_{j+1}. In questo stato le proposizioni $e_0, ..e_m$ sono verificate. Una volta completati gli accertamenti medici, la visita si conclude passando dallo stato s_{j+1} allo stato $sj + 2$ e qui la proposizione h_i è verificata.
- Ogni visita oncologica è modellata come un percorso, e quindi nel rispetto del preordine P, la visita i-ma sarà un percorso che verifica tutte le proprietà che verificano anche le visite precedenti, individuate come percorsi minori del percorso i-mo nel modello $\mathcal{M}$. Inoltre, all'inizio di una generica visita i, sono verificate le proposizioni $h_0, .., h_{i-1}$ in quanto all'inizio dell'osservazione il medico possiede l'informazione che tutte le visite precedenti a quella attuale sono andate a buon fine (vedi **figura 4.1**).

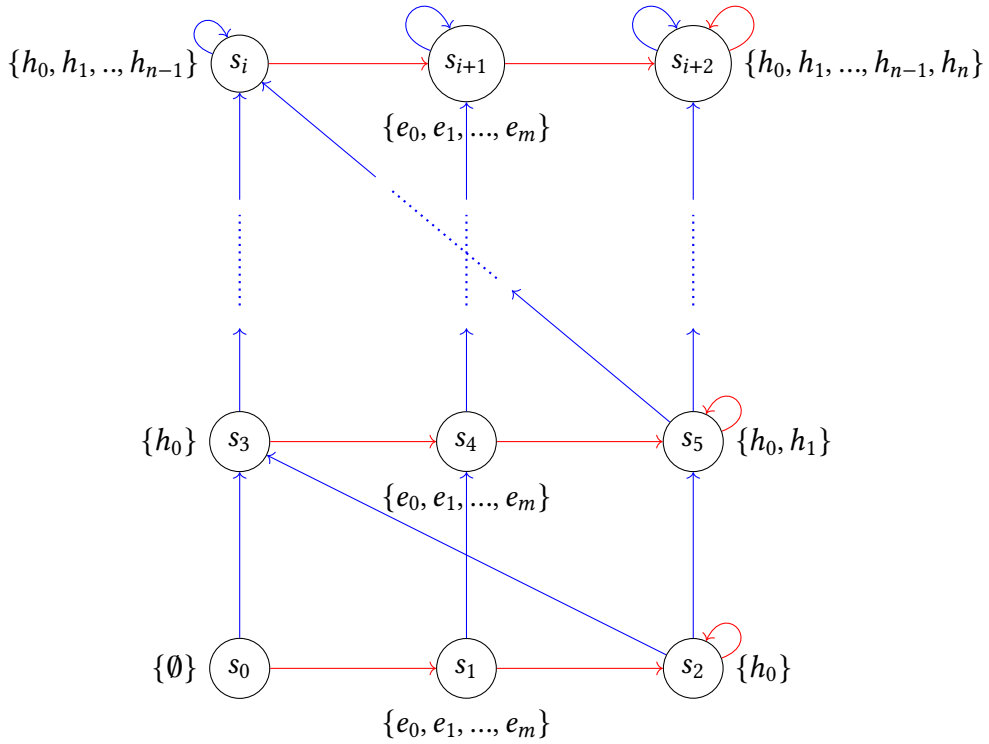


Figure 4.1: Modello di visite oncologiche effettuate su un paziente

Pertanto, questo modello non solo ci consente di modellare l'aspetto temporale di una visita medica, ma ci permette anche di **tracciare** come, tra le varie visite, venga conservato lo storico delle visite precedenti. Poiché gli stati sono finiti, non è possibile avere un percorso più lungo dell'ultimo definito in figura (se non il percorso stesso). Di conseguenza, se un paziente risulta sano, lo sarà costantemente durante tutte le osservazioni del modello. Quindi, per verificare lo stato di salute del paziente sarà sufficiente verificarlo al percorso più in alto nel rispetto del preordine P. Il modello soddisfa la seguente formula: $AG(\bigwedge_{i=0}^{m} e_i \rightarrow AX \bigvee_{j=0}^{n} h_j) \rightarrow \neg C$, cioè "Se, globalmente, ogni volta che vengono effettuati gli accertamenti oncologici, ciò porta sempre ad uno stato successivo in cui almeno una visita medica ha dato esito positivo, allora ciò implica che non è stata rilevata alcuna massa tumorale sul paziente".
Nella teoria dei giochi, la presenza o l'assenza di memoria gioca un ruolo cruciale nella complessità dei problemi e nella loro risolubilità. La memoria, in questo contesto, si riferisce alla capacità dei giocatori di ricordare le mosse precedenti e le informazioni passate durante il gioco. Quando i giocatori hanno memoria, possono basare le loro decisioni su tutta la storia delle mosse precedenti. Questo permette strategie più sofisticate e articolate, ma allo stesso tempo può rendere il problema di trovare una strategia vincente più complesso. In alcuni casi, la capacità di ricordare può trasformare un problema della verifica di proprietà da

decidibile in uno indecidibile, poiché la quantità di informazioni da considerare può crescere esponenzialmente con il numero di mosse. Nel caso preso in esame stiamo considerando un'alternativa per rappresentare la memoria e la conoscenza parziale di un agente, pertanto la verifica di una proprietà temporale in questo modello come "durante ogni osservazione è sano", è sufficiente guardare la prima e l'ultima osservazione e considerare due casi:

  - se il paziente è sano in entrambe le osservazioni si può concludere;
  - se il paziente è sano alla prima osservazione ma risulta malato all'ultima osservazione, allora è necessario fare una ricerca binaria sulle osservazioni per individuare la corretta osservazione in cui il paziente ha riscontrato la patologia.

- **Scrittura su un database - informazione tracciabile e persistente**. Per questo scenario, possiamo adottare un modello temporale che definisce gli stati e le transizioni in cui un numero fissato di utenti può effettuare operazioni di lettura/scrittura su un database. Anche in questo caso, considerando n osservazioni, durante l'osservazione i-esima è possibile tenere traccia di tutte le operazioni di scrittura effettuate nelle osservazioni precedenti. Quindi, in ogni istante di osservazione si saprà sempre chi ha eseguito una scrittura sul database e quale dato ha inserito. Il modello temporale corrispondente a una singola osservazione può essere definito nel modo seguente:

  1. Stato "init" in cui il modello è in attesa di una richiesta di lettura/scrittura;
  2. Stato "write" per la scrittura;

  Inoltre, si possono considerare la seguente proposizioni atomica:

  1. w_{ij}: l'utente i-esimo ha scritto il dato j.

  Ogni osservazione consente di tenere traccia di ciò che gli utenti hanno scritto in precedenza attraverso il preordine P. In particolare, per un'osservazione i-esima, lo stato di "write" dell'osservazione corrente è in preordine con lo stato "init" dell'osservazione successiva: $write_i P init_{i+1}$. In questo modo, se si vuole verificare se un determinato utente ha scritto uno specifico dato nel database, sarà sufficiente verificarlo a partire dall'ultima osservazione del modello (**figura 4.2**).

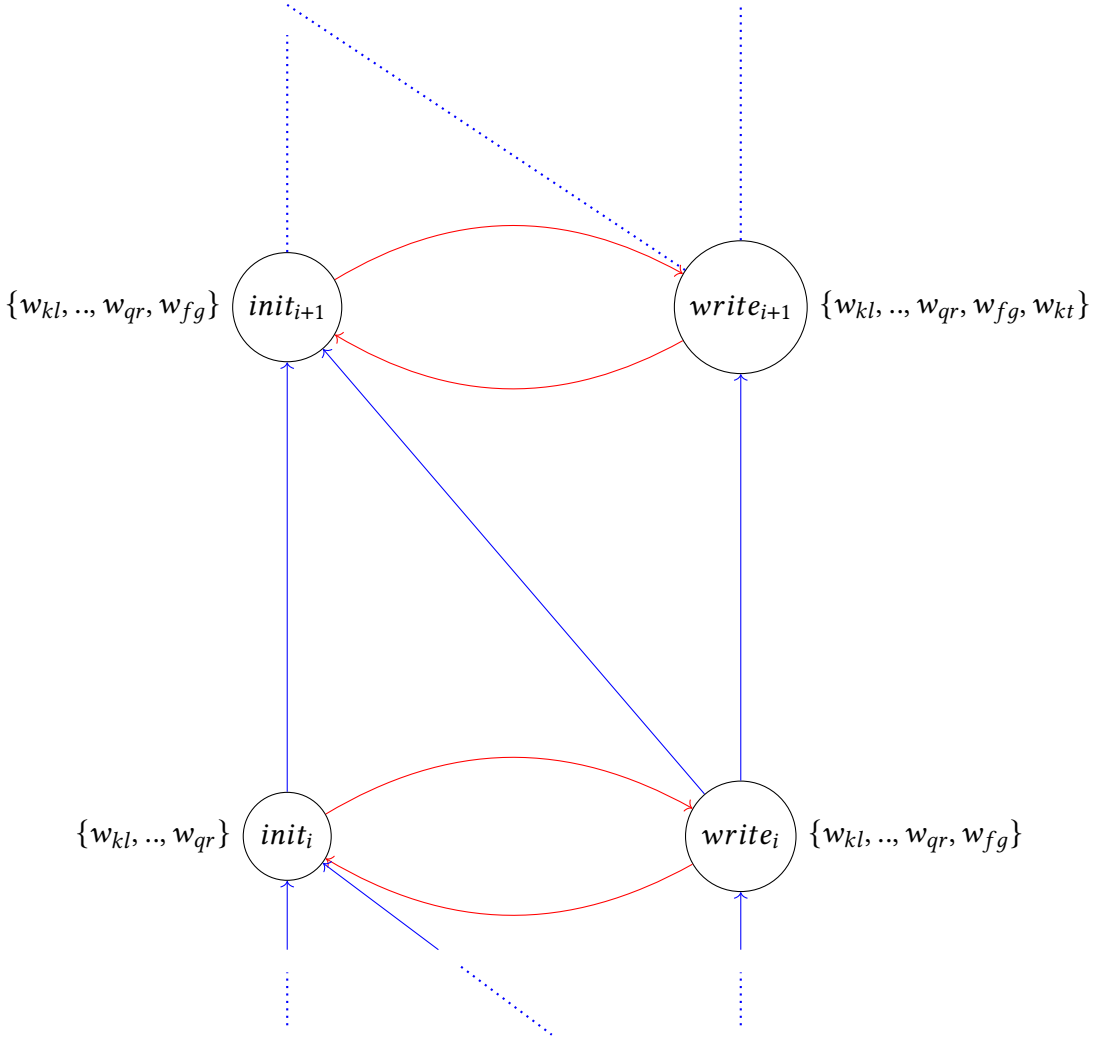


Figure 4.2: Modello di tracciabilità di un database

In questo modo si può garantire che non avvenga mai una sovrascrittura oppure, nel caso in cui il modello preveda operazioni di sovrascrittura, sarà sempre possibile sapere quale dato è stato sovrascritto, poiché le informazioni acquisite sul modello osservato non possono essere eliminate in alcun modo.
Estendendo ulteriormente questo concetto, è possibile applicare opportuni modelli temporali anche ai database distribuiti. Questo approccio permetterebbe di tenere traccia delle transazioni su scala più ampia, migliorando la gestione e la verifica delle operazioni in ambienti distribuiti. Ad esempio, ogni nodo del database distribuito potrebbe mantenere un registro temporale locale delle operazioni di lettura e scrittura, sincronizzandosi periodicamente con gli altri nodi per garantire la coerenza globale dei dati. Questi modelli temporali potrebbero offrire un'alternativa robusta e tracciabile ai metodi attualmente utilizzati per monitorare le transazioni nei database, facilitando la sicurezza, l'integrità dei dati e soprattutto la sostenibilità

ambientale.
La capacità di tracciare l'ordine delle operazioni nel tempo e di identificare chiaramente chi ha modificato cosa, e quando, rappresenta un vantaggio significativo per la gestione di database complessi e ad alta intensità di transazioni.

- **Monitoraggio del terreno agricolo** Analogamente allo scenario del medico, è possibile tracciare lo stato di salute di un terreno agricolo adottando un modello temporale che definisce gli stati e le transizioni in cui si effettuano varie operazioni di monitoraggio e manutenzione. In questo caso, considerando n osservazioni, durante l'osservazione i-esima è possibile tenere traccia di tutte le attività svolte nelle osservazioni precedenti. Quindi, in ogni istante di osservazione, si saprà sempre quali interventi sono stati effettuati sul terreno e quali condizioni sono state rilevate.
  Il modello temporale per il monitoraggio di un terreno agricolo potrebbe includere stati come:

  1. stato *init* in cui il modello è in attesa di una richiesta di monitoraggio o intervento;
  2. stato *monitor* per il rilevamento di parametri del terreno come umidità, pH, nutrienti, e presenza di parassiti;
  3. stato *intervento* per le operazioni di manutenzione come irrigazione, fertilizzazione e applicazione di pesticidi.

  Le proposizioni atomiche potrebbero essere definite come:

  1. m_j: il parametro j del terreno è stato monitorato;
  2. in_j: è stata effettuata l'operazione di manutenzione j.

  Ogni osservazione consente di tenere traccia delle condizioni e degli interventi sul terreno attraverso il preordine P. Ad esempio, per un'osservazione i-esima, lo stato di *intervene* dell'osservazione precedente è in preordine con lo stato *init* dell'osservazione corrente: $intervene_{i-1} P init_i$. In questo modo, se si vuole verificare se un determinato intervento è stato effettuato o se un certo parametro del terreno è stato monitorato, sarà sufficiente analizzare le osservazioni precedenti a partire dalla prima.
  Estendendo ulteriormente questo concetto, è possibile integrare tecnologie avanzate come sensori IoT e droni per il monitoraggio continuo e dettagliato del terreno. I sensori possono raccogliere dati in tempo reale sui parametri del suolo e delle colture, mentre i droni possono fornire immagini aeree per l'analisi visiva delle condizioni del terreno.
  Questo approccio non solo migliora la gestione e la manutenzione del terreno agricolo, ma permette anche una reazione tempestiva ai cambiamenti delle condizioni ambientali, ottimizzando le risorse e aumentando la produttività. La capacità di

tracciare l'ordine e l'efficacia delle operazioni nel tempo offre un vantaggio significativo per l'agricoltura di precisione, consentendo decisioni informate e basate sui dati per migliorare la salute del terreno e la qualità delle colture.

- **Tracking di autenticazione**. Per questo scenario, possiamo adottare un modello temporale che definisce gli stati e le transizioni in cui un utente ha a disposizione diversi metodi di autenticazione, come password, biometrie, token, e autenticazione a due fattori (2FA). Il modello include uno stato per l'autenticazione fallita e uno per l'autenticazione avvenuta con successo. Considerando n osservazioni, durante l'osservazione i-esima, è possibile tenere traccia dell'azione compiuta dall'utente, ovvero quale metodo di autenticazione ha utilizzato e se l'autenticazione è fallita o riuscita. Di conseguenza, in ogni momento di osservazione, sarà sempre noto quale azione l'utente ha intrapreso per accedere al sistema e se ha commesso un errore.
  Inoltre, grazie al preordine, è possibile stabilire esattamente l'ordine delle azioni compiute dall'utente. Ogni osservazione terrà traccia temporale di una e una sola azione effettuata dall'utente, permettendo di ricostruire con precisione la sequenza di tentativi di autenticazione. Questo approccio consente di analizzare i pattern di comportamento dell'utente, identificando, ad esempio, quali metodi di autenticazione sono più frequentemente utilizzati o quali causano più frequentemente fallimenti.
  Estendendo ulteriormente questo modello, possiamo incorporare meccanismi di sicurezza avanzati che reagiscono dinamicamente in base al comportamento dell'utente. Ad esempio, se un utente ha ripetuti fallimenti di autenticazione, il sistema potrebbe richiedere metodi di autenticazione più rigorosi o notificare l'amministratore di sistema di potenziali attività sospette come bruteforce. Inoltre, possiamo integrare l'analisi del contesto, come la posizione geografica o il dispositivo utilizzato, per migliorare la precisione e la sicurezza del processo di autenticazione.
  Questo modello non solo migliora la sicurezza del sistema, ma anche l'esperienza dell'utente, offrendo un accesso più fluido e personalizzato basato sulle sue abitudini e comportamenti di autenticazione.

- **Machine learning model che classifica nuovi dati nel tempo**. Per questo scenario, possiamo adottare un modello temporale che definisce gli stati e le transizioni in cui un classificatore viene addestrato utilizzando un algoritmo di machine learning generico o una rete neurale. Considerando n osservazioni, durante l'osservazione i-esima, è possibile tenere traccia di tutto ciò che il modello ha già imparato a classificare nelle osservazioni precedenti. Di conseguenza, in ogni momento di osservazione, sarà sempre noto ciò che il modello è già in grado di classificare.
  Un altro scenario simile potrebbe riguardare un large language model (LLM) che, acquisendo progressivamente più contesto da nuovi prompt, riesce a generare nuovi testi, immagini o altri contenuti multimediali. In questo caso, il modello mantiene traccia di ciò che è già capace di generare a ogni nuova osservazione.
  Estendendo ulteriormente questo concetto, possiamo implementare meccanismi che consentono al modello di aggiornarsi dinamicamente in base ai nuovi dati ricevuti.

Ad esempio, ogni nuova osservazione potrebbe includere non solo informazioni su ciò che il modello ha imparato, ma anche feedback sulla sua accuratezza, permettendo un miglioramento continuo. Questo feedback può provenire da utenti finali o da un sistema di valutazione automatizzato che verifica la correttezza delle classificazioni o delle generazioni.
Inoltre, possiamo introdurre un componente di adattamento contestuale, che permette al modello di modificare le sue previsioni in base al contesto specifico dell'osservazione corrente. Ad esempio, un classificatore potrebbe migliorare le sue performance in domini specifici man mano che acquisisce più dati relativi a quel dominio. Analogamente, un LLM potrebbe affinare la qualità dei suoi output comprendendo meglio le esigenze e le preferenze degli utenti attraverso l'interazione continua.
Questo approccio non solo renderebbe il modello più robusto e accurato, ma lo renderebbe anche più adattabile e personalizzabile, offrendo risultati che migliorano costantemente con l'aumentare dei dati e delle interazioni.

4.3 Implementazione in Vitamin

VITAMIN [14] è un prototipo innovativo, progettato per risolvere le complessità associate alla verifica dei Sistemi Multi-Agente (MAS). Il suo scopo principale è di facilitare la verifica formale dei MAS in un modo che sia modulare e versatile. A differenza delle metodologie e dei framework di verifica esistenti per i MAS, che spesso sono rigidi e difficili da usare, VITAMIN è stato sviluppato con l'obiettivo di essere facilmente estendibile. Questo significa che può adattarsi per accogliere una varietà di logiche, che sono utilizzate per definire le proprietà da verificare, e di modelli, che stabiliscono su cosa tali proprietà devono essere verificate. A tale scopo, durante la stesura dell'elaborato è stato possibile estendere questo tool per il supporto ad $ICTL_{C_3}$.
La **figura 4.3** illustra i principali componenti del tool, evidenziando la separazione tra logiche e modelli, fondamentale per permettere un'evoluzione indipendente e flessibile in entrambe le direzioni. Questo approccio facilita la verifica di diverse logiche su modelli distinti.
Un aspetto rilevante è l'interfaccia dedicata alla gestione della verifica formale, indipendente e altamente modulare. La verifica viene eseguita in vari modi. La metodologia di interesse per l'elaborato è la verifica esplicita basata su punto fisso (CTL).
Il framework prevede tre categorie di utenti: l'utente finale, che utilizza il tool senza modificarlo; gli sviluppatori di alto livello, che estendono o introducono nuovi modelli e logiche; e gli sviluppatori di basso livello, che ottimizzano le implementazioni. Gli sviluppatori di alto livello si concentrano sulla definizione e verifica dei modelli e delle logiche, mentre quelli di basso livello migliorano l'implementazione con tecniche di ottimizzazione.
L'Interfaccia del Model Checker, sviluppata dagli sviluppatori di basso livello, consente agli utenti finali e agli sviluppatori di alto livello di utilizzare facilmente le ottimizzazioni. L'architettura del tool presenta punti di forza come la modularità, la facilità d'uso, la

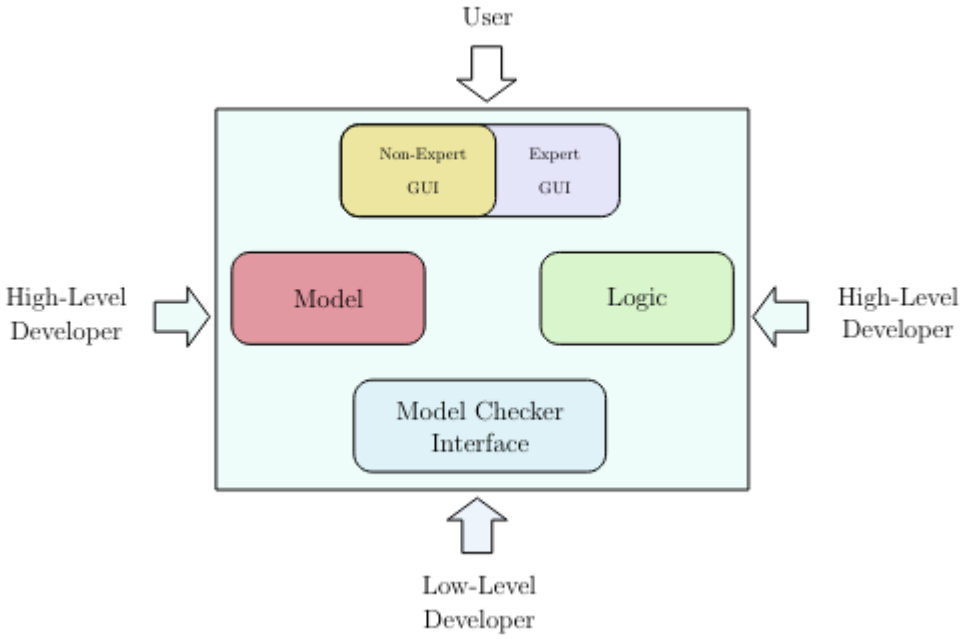


Figure 4.3: Vitamin Architecture

possibilità di distribuzione dei componenti di verifica e una documentazione completa per supportare utenti e sviluppatori.
L'input del modello per la logica è fornito tramite un file di testo. Questo file è suddiviso in sezioni e ciascuna sezione viene opportunamente gestita da un modulo di input processing. Le sezioni contengono le seguenti informazioni:

- **Transition**: una matrice quadrata in cui la posizione i,j denota la relazione tra lo stato s_i ed s_j. Essendo che il modello è birelazionale, una relazione tra due stati potrà essere di preordine P, di transizione R, entrambe (separate da una virgola e denotate come "P,R") oppure nessuna (denotata con "0");
- **Name_State**: una lista di nomi di stati;
- **Initial_State**: denota lo stato iniziale
- **Atomic_Proposition**: una lista di nomi di proposizioni atomiche
- **Labeling**: Una funzione di etichettatura rappresentata come una matrice booleana. se in posizione i,j c'è un 1 allora significa che nello stato s_i è vera la proposizione j-ma nella lista dei nomi delle proposizioni atomiche.

Di seguito un esempio di file in input:

model.txt

```
Transition
P R 0 P 0 0
R P,R R 0 P 0
0 R P P 0 P
0 0 0 P R 0
0 0 0 R P R
0 0 0 R R P
Name_State
s0 s1 s2 s3 s4 s5
Initial_State
s0
Atomic_propositions
e h c
Labelling
0 0 0
1 0 0
0 1 0
0 1 0
1 0 0
0 1 0
```

Una volta che il modello viene letto, l'algoritmo verifica che soddisfi tutti i vincoli di un modello birelazionale. In particolare, si assicura che il preordine P sia una relazione binaria che rispetti le proprietà di riflessività, antisimmetria e transitività. Inoltre, controlla che la relazione di transizione R sia seriale e che il modello aderisca alle condizioni C_1, C_2, C_3. Infine, l'algoritmo verifica che la funzione di etichettatura sia definita correttamente in accordo con il preordine specificato. L'algoritmo di model checking per ICTL segue lo sviluppo di CTL, con adattamenti per gestire correttamente gli operatori $\neg$ e $\rightarrow$, come descritto nel capitolo precedente. Infine è stato sviluppato un algoritmo per generare dei modelli che rispettano la struttura specificata nei casi d'uso della sezione precedente, in cui è possibile definire un numero arbitrario di stati "colonna" che rappresentano l'evoluzione temporale del modello e un numero arbitrario di stati "riga" che rappresentano il numero di osservazioni. Dopo la generazione del modello, l'algoritmo verifica che la struttura sia un modello birelazionale che rispetta tutte le condizioni di cui sopra.

Per verificare l'efficienza del modello sono stati effettuati dei benchmark nel seguente modo:

1. Generazione del modello con un numero arbitrario di righe e di colonne i cui valori sono specificati nelle tabelle di seguito; [4]. Viene riportato il numero di relazioni tra stati in quanto queste sono generate in numero casuale;
2. Processing per calcolare archi di transizioni, di preordine e chiusure per evitare di doverle calcolare più volte;
3. Model checking della proprietà $AG(e \longrightarrow EFh) \longrightarrow !c$ che risulta sempre vera in ogni caso di test.
4. Per ciascun caso sono riportati i tempi di esecuzione di ogni fase per un totale di 10 esecuzioni per ogni caso.

I benchmark sono effettuati su un PC desktop con sistema operativo **Ubuntu 24.04** dotato di una CPU **Intel i7 12700k** e **64 GB** di **RAM**. I risultati del benchmark sono riportati nelle tabelle di seguito:

Grid	States	K	Relations	T_Gen (Sec)	T_Proc (Sec)	T_MC (Sec)
10 x 10	100	1	1186	0.0028	0.0037	0.0037
		2	849	0.0011	0.0020	0.0034
		3	1000	0,0016	0.0017	0.0036
		4	1180	0.0027	0.0019	0.0035
		5	776	0.0008	0.0017	0.0034
		6	1047	0.0019	0.0018	0.0040
		7	856	0.0010	0.0017	0.0033
		8	1042	0.0019	0.0019	0.0037
		9	935	0.0013	0.0018	0.0035
		10	1195	0.0028	0.0018	0.0036
20 x 20	400	1	8116	0.0201	0.0225	0.0186
		2	8005	0.0175	0.0266	0.0209
		3	8216	0.0171	0.0220	0.0219
		4	8376	0.0181	0.0222	0.0195
		5	9150	0.0227	0.0220	0.0218
		6	8199	0.0170	0.0220	0.0215
		7	6179	0.0077	0.0214	0.0204
		8	7062	0.0112	0.0260	0.0192
		9	9140	0.0227	0.0215	0.0183
		10	9420	0.0246	0.0223	0.0193

Table 4.1: Model Checking ICTL Benchmark 1

[4] La griglia di stati nell'esperimento è sempre rappresentata come una matrice quadrata, per semplificare la definizione del numero di test case.

Grid	States	K	Relations	T_Gen (Sec)	T_Proc (Sec)	T_MC (Sec)
30 x 30	900	1	29627	0.0756	0.1052	0.052
		2	23908	0.0444	0.1047	0.048
		3	16959	0.0217	0.1014	0.037
		4	25627	0.0514	0.1076	0.046
		5	21449	0.0335	0.1015	0.045
		6	31352	0.0862	0.1093	0.056
		7	21440	0.0345	0.1078	0.045
		8	20706	0.0298	0.1022	0.044
		9	25602	0.0520	0.1054	0.047
		10	30410	0.0772	0.1069	0.051
40 x 40	1600	1	39696	0.0480	0.3058	0.080
		2	68816	0.1830	0.3129	0.119
		3	41182	0.0563	0.3056	0.077
		4	62471	0.1373	0.3105	0.109
		5	45284	0.0723	0.3096	0.088
		6	38216	0.0416	0.3106	0.073
		7	66500	0.1720	0.3034	0.110
		8	51395	0.0918	0.3086	0.093
		9	56921	0.1121	0.3086	0.100
		10	63455	0.1490	0.3071	0.106
50 x 50	2500	1	139860	0.4084	0.7205	0.252
		2	132487	0.3543	0.7436	0.237
		3	121501	0.2956	0.7458	0.212
		4	89755	0.1627	0.7400	0.160
		5	109566	0.2391	0.7345	0.191
		6	95645	0.1810	0.7348	0.170
		7	124191	0.3055	0.7398	0.216
		8	135926	0.3775	0.7490	0.244
		9	136032	0.3807	0.7475	0.243
		10	107887	0.2294	0.7383	0.190
60 x 60	3600	1	244977	0.7515	1.4586	0.487
		2	227399	0.6199	1.4909	0.452
		3	239472	0.7125	1.4753	0.475
		4	162400	0.3206	1.4716	0.298
		5	128907	0.2134	1.4423	0.227
		6	240841	0.7196	1.4960	0.488
		7	147976	0.2767	1.4492	0.266
		8	165047	0.3300	1.4679	0.308
		9	242334	0.7308	1.4748	0.480
		10	218553	0.5724	1.4882	0.432

Table 4.2: Model Checking ICTL Benchmark 2

Grid	States	K	Relations	T_Gen (Sec)	T_Proc (Sec)	T_MC (Sec)
70 x 70	4900	1	362994	1.0535	2.7744	0.764
		2	383601	1.1883	2.7984	0.841
		3	289721	0.6786	2.7638	0.591
		4	392967	1.2491	2.7999	0.881
		5	360808	1.0337	2.7982	0.767
		6	190790	0.3265	2.7255	0.335
		7	254451	0.5407	2.7614	0.501
		8	328735	0.8573	2.7514	0.683
		9	238931	0.4840	2.7735	0.458
		10	254396	0.5519	2.7430	0.499
80 x 80	6400	1	375773	0.8577	4.5756	0.783
		2	557509	1.7556	4.6946	1.304
		3	563252	1.7764	4.6640	1.323
		4	344900	0.7496	4.6333	0.691
		5	566209	1.7941	4.6583	1.343
		6	475680	1.2756	4.6437	1.085
		7	544360	1.6581	4.7111	1.271
		8	355620	0.7836	4.6106	0.723
		9	475543	1.2772	4.6722	1.073
		10	322878	0.6813	4.6276	0.616
90 x 90	8100	1	477984	1.1124	7.2530	0.989
		2	655080	1.8072	7.3624	1.594
		3	736069	2.2252	7.4695	1.843
		4	755217	2.3438	7.3908	1.907
		5	774249	2.4533	7.3967	1.976
		6	549869	1.3778	7.3515	1.225
		7	719078	2.1180	7.4387	1.789
		8	735557	2.2219	7.3820	1.839
		9	602642	1.5581	7.3723	1.393
		10	814150	2.7141	7.4902	2.128
100 x 100	10000	1	998355	3.1354	11.248	2.648
		2	950732	2.8345	11.207	2.469
		3	988561	3.0883	11.212	2.607
		4	104717	3.4141	11.335	2.877
		5	112637	3.9884	11.244	3.098
		6	860342	2.4584	11.184	2.181
		7	666571	1.7083	10.869	1.471
		8	998562	3.1352	11.365	2.653
		9	961510	2.9117	11.223	2.575
		10				

Table 4.3: Model Checking ICTL Benchmark 3

Grid	States	K	Relations	T_Gen (Sec)	T_Proc (Sec)	T_MC (Sec)
110 x 110	12100	1	1041114	3.0092	15.788	2.667
		2	1500205	5.3706	16.010	4.511
		3	1109175	3.2643	15.932	2.933
		4	1304733	4.1712	16.299	3.720
		5	1471506	5.2614	16.336	4.383
		6	1304401	4.1697	16.343	3.729
		7	1388107	4.6701	16.321	4.047
		8	1388375	4.6818	16.374	4.081
		9	889250	2.4420	16.237	2.071
		10	1230546	3.7921	16.273	3.450
120 x 120	14400	1	1505308	4.6629	22.368	4.410
		2	1399921	4.2827	22.488	3.958
		3	1369180	4.1763	22.600	3.812
		4	1358751	4.0886	22.585	3.756
		5	1204423	3.4898	22.547	3.099
		6	1964286	7.3062	22.745	6.446
		7	1969823	7.4515	22.737	6.469
		8	1855024	6.5803	22.739	5.958
		9	1921106	7.1166	22.776	6.276
		10	1790336	6.2598	22.739	5.708
130 x 130	16900	1	1528758	4.7169	30.893	4.176
		2	1771322	5.6479	31.001	5.353
		3	2135457	7.4002	31.365	7.098
		4	1710899	5.4887	31.330	5.074
		5	2070729	6.9936	31.092	6.807
		6	2079080	7.0766	31.144	6.871
		7	2502609	9.7341	31.246	8.898
		8	1458264	4.5444	30.979	3.863
		9	2204540	7.7160	31.540	7.446
		10	2237703	7.9673	31.420	7.620
140 x 140	19600	1	2349299	8.0316	41.634	7.849
		2	1562757	5.0525	41.341	3.709
		3	2030248	6.6880	42.144	6.228
		4	2597621	9.3169	41.886	9.016
		5	3095448	12.252	42.459	11.81
		6	1905777	6.1482	41.268	5.568
		7	3042461	11.954	42.200	11.55
		8	2257254	7.6391	41.650	7.411
		9	1562827	5.0214	41.932	3.690
		10	3058585	11.912	41.797	11.61

Table 4.4: Model Checking ICTL Benchmark 4

Grid	States	K	Relations	T_Gen (Sec)	T_Proc (Sec)	T_MC (Sec)
150 x 150	22500	1	3431276	12.839	54.948	13.14
		2	3675188	14.592	55.105	14.54
		3	1754626	6.2307	54.754	3.783
		4	3627917	14.205	55.070	14.30
		5	1842593	6.5862	55.278	4.425
		6	2586505	8.9804	54.600	8.586
		7	2793972	9.9102	55.111	9.659
		8	3507199	13.507	55.007	13.61
		9	3702417	14.890	55.590	14.67
		10	2700185	9.3464	54.821	9.211
160 x 160	25600	1	4407663	17.502	70.209	18.36
		2	3040934	11.244	72.092	10.40
		3	3936809	15.093	71.083	15.51
		4	3591322	13.132	72.179	13.60
		5	2961151	10.825	70.102	9.902
		6	4612183	19.131	71.932	19.58
		7	3442806	12.600	70.189	12.70
		8	2645273	9.6659	70.774	8.109
		9	2732794	9.6840	69.938	8.607
		10	2273774	8.2265	70.615	5.845
170 x 170	28900	1	4145564	15.604	92.480	16.11
		2	3865936	14.804	91.944	14.48
		3	3133762	11.786	91.616	9.929
		4	5435032	23.029	91.878	24.25
		5	4239821	15.962	90.214	16.68
		6	5364408	22.373	90.255	23.70
		7	5339457	21.900	90.230	23.53
		8	5057569	20.668	91.001	21.88
		9	5546005	23.609	90.326	24.95
		10	4750782	18.493	90.254	19.93
180 x 180	32400	1	5736420	23.221	113.71	26.12
		2	6230738	26.822	113.15	29.60
		3	3324939	13.171	112.17	9.598
		4	6354195	26.726	113.58	30.41
		5	5995091	24.779	115.11	27.94
		6	5186293	20.736	114.85	22.33
		7	3706237	14.617	113.18	12.33
		8	6300050	27.098	114.60	30.00
		9	5634644	22.614	113.55	25.45
		10	6071043	25.485	113.27	28.50

Table 4.5: Model Checking ICTL Benchmark 5

Grid	States	K	Relations	T_Gen (Sec)	T_Proc (Sec)	T_MC (Sec)
190 x 190	36100	1	6053876	16.696	139.12	12.96
		2	7776998	23.203	141.64	25.38
		3	9621183	33.513	140.60	38.27
		4	9058517	30.343	140.44	34.70
		5	8847514	28.822	140.35	33.20
		6	9176243	31.004	140.40	35.57
		7	6499929	18.283	139.69	16.25
		8	7055026	20.093	139.57	20.29
		9	6248603	17.601	140.84	14.48
		10	7606588	22.276	140.05	24.15
200 x 200	40000	1	9004628	40.449	172.48	46.57
		2	8958451	40.457	173.84	46.54
		3	7988797	34.885	172.35	39.09
		4	7166129	30.536	171.53	33.04
		5	6732860	28.296	171.49	29.88
		6	5843599	24.931	171.00	23.58
		7	4354216	19.670	172.39	12.79
		8	7907530	34.179	171.89	38.50
		9	5557490	23.318	171.09	21.58
		10	5812107	24.586	171.56	23.42
210 x 210	44100	1	8104010	34.966	206.08	38.65
		2	6428162	27.828	212.39	26.03
		3	6356365	27.447	208.73	25.34
		4	6674755	28.993	212.25	27.85
		5	8657578	37.695	206.91	42.81
		6	9338724	41.070	209.40	48.08
		7	7642919	32.912	206.50	35.03
		8	10475476	48.691	212.49	56.75
		9	8212752	35.083	209.03	39.30
		10	5793735	25.649	212.02	21.41
220 x 220	48400	1	9047011	39.259	249.21	44.05
		2	9934484	43.813	251.62	51.24
		3	6689900	31.061	248.11	25.65
		4	9386127	41.747	252.12	46.95
		5	8854648	38.328	257.70	42.63
		6	11058442	50.448	256.53	60.50
		7	9172759	40.544	258.79	45.27
		8	8451127	37.524	253.77	:39.3
		9	11446365	52.481	253.43	64.96
		10				

Table 4.6: Model Checking ICTL Benchmark 6

Grid	States	K	Relations	T_Gen (Sec)	T_Proc (Sec)	T_MC (Sec)
230 x 230	52900	1	10765067	47.691	289.53	57.07
		2	8344770	39.489	299.42	36.54
		3	6805984	33.167	300.10	23.20
		4	13605996	65.530	299.96	82.99
		5	10464207	48.451	310.13	55.88
		6	9934713	45.314	294.52	48.83
		7	11571972	52.906	299.85	65.05
		8	8474210	39.634	293.22	36.61
		9	12366143	57.562	299.98	72.70
		10	12262516	57.783	303.20	69.01

Table 4.7: Model Checking ICTL Benchmark 7

Conclusioni e Sviluppi Futuri

Lo scopo di questa tesi è stato quello di presentare una versione intuizionista della logica temporale CTL evidenziandone le principali proprietà che la costituiscono ed esibendo un algoritmo di model checking efficiente. Attraverso l'analisi e la discussione, abbiamo dimostrato come questa versione intuizionista possa essere particolarmente utile in contesti in cui un modello acquisisce informazioni durante diverse osservazioni e inoltre, abbiamo illustrato come tale approccio sia in grado di delineare efficacemente una cronologia relativa all'ordine di acquisizione dei dati in termini di azioni compiute da un utente/agente, uno storico di rilevazioni su un individuo o un oggetto oppure per tracciare efficacemente la provenienza di un dato.
Sono state esplorate le potenzialità di ICTL, fornendo una comprensione più profonda del suo funzionamento e delle sue applicazioni. Questo lavoro contribuisce alla letteratura esistente su CTL, offrendo una nuova prospettiva e aprendo la strada a ulteriori ricerche in questo campo.
È stato successivamente esteso il tool VITAMIN Model Checker implementando l'algoritmo di model checking ICTL. Sulla base dei risultati ottenuti, è possibile concludere che l'algoritmo risulta essere molto efficiente anche in presenza di un numero elevato di stati e relazioni. Questo implica che la soluzione proposta è altamente scalabile, mantenendo prestazioni elevate anche con dati di grandi dimensioni.
Si spera che i risultati presentati in questa tesi possano stimolare ulteriori discussioni e ricerche sulle logiche temporali e i sistemi multi-agente.
Guardando al futuro, ci sono molte opportunità per ulteriori sviluppi e applicazioni a partire da ICTL:

- **Model checking ICTL**. Durante la stesura dell'elaborato è stato possibile fornire un algoritmo di model checking per $ICTL_{C_3}$. Questo deriva dal fatto che sotto questa condizione, le equivalenze del punto fisso per gli operatori $A\varphi_1 \mathcal{U} \varphi_2$ e $A\varphi_1 \mathcal{R} \varphi_2$ restano valide. Questo non accade per ICTL senza la condizione C_3. Sarebbe interessante poter sviluppare un algoritmo di model checking anche per questa versione. Da notare che il frammento $IECTL_{C_3}$ e il frammento $IECTL$ sono equivalenti. Questo frammento include tutti gli operatori di ICTL esclusi quelli che quantificano universalmente sui percorsi. La verifica di formule del tipo $AX\varphi$ risulta abbastanza intuitiva in quanto è stata dimostrata la **proposizione 11**. Bisognerebbe trovare una procedura efficiente per la verifica di proprietà del tipo $A\varphi_1 \mathcal{U} \varphi_2$ e $\varphi_1 \mathcal{R} \varphi_2$;

- **Model Checking ILTL**. Come già è stato accennato, attualmente esistono delle versioni intuizioniste di LTL [41]. In particolare possiamo trovare la logica ITL^e la quale risulta godere delle proprietà dei modelli finiti, per cui questa logica è decidibile ma la procedura di decisione si pensa sia almeno $NON - ELEMENTARY$. ITL^p invece non godendo di queste proprietà non è detto nemmeno sia decidibile. Inoltre ci sono stati alcuni tentativi di assiomatizzazione per ITL^e ma risulta difficile trovarne una corretta e completa [50], [51]. Risulta quindi una sfida molto stimolante riuscire a trovare un algoritmo di model checking efficiente per questa logica. Come conseguenza, sarebbe possibile estendere la logica intuizionista a logiche temporali più espressive come CTL* in quanto classicamente, il model checking CTL* è implementato mediante una combinazione dei due algoritmi di model checking CTL ed LTL;

- **Model Checking ICTL^2**. L'enorme divario tra la complessità della verifica di modelli CTL e LTL ha portato allo sviluppo di strumenti di verifica di modelli per CTL, mentre i model checker per LTL sono rimasti indietro. Tuttavia, gli utenti di questi strumenti devono confrontarsi con la limitata capacità espressiva di CTL e sono spesso costretti a rinunciare a verificare molti comportamenti importanti come i vincoli di fairness. Di conseguenza, trovare linguaggi di specifica che siano più espressivi di CTL e che allo stesso tempo mantengano la sua complessità di verifica dei modelli è un problema impegnativo ed è stato un attivo campo di ricerca. In [52], è stato proposto il linguaggio, CTL^2, il quale è il risultato di un nuovo approccio per definire sotto-linguaggi di CTL*. Questo approccio consente un numero limitato di operatori di tempo lineare all'interno delle formule di percorso di CTL*. Nell'articolo viene mostrato che, oltre all'aumento del potere espressivo, un notevole vantaggio di CTL^2 è la presentazione chiara e intuitiva che fornisce per specifiche le cui equivalenze CTL sono complicate e molto difficili da comprendere. Il model checking CTL^2 ha una complessità lineare sia nella formula sia nel modello da verificare, esattamente come quello per CTL. Sarebbe quindi interessante, data l'importanza e un maggiore utilizzo di questo tipo di formule rispetto a formule CTL*, svilupparne una versione intuizionista;

- **Estensione ad altre logiche**. Sarebbe naturale estendere questa interpretazione intuizionista alla versione epistemica di CTL, cioè CTLK [2], ad ATL (Alternating Time Temporal Logic) [53] e alla sua versione epistemica ATEL [54] (Alternating Time and Epistemic Temporal Logic). Infine si potrebbe proporre anche una versione intuizionista di Strategy Logic (SL) [55], [56].

- **Altre proprietà di ICTL**. Sarebbe affascinante approfondire ulteriormente ICTL, esplorando delle proprietà aggiuntive. Ad esempio, si potrebbe cercare di trovare una traduzione da CTL a ICTL, che dimostrerebbe l'equiconsistenza logica tra le due.

Bibliography

[1] Edmund M Clarke and Jeannette M Wing. "Formal methods: State of the art and future directions". In: *ACM Computing Surveys (CSUR)* 28.4 (1996), pp. 626–643.

[2] Ronald Fagin, JY Halpern, Y Moses, and MY Vardi. "Reasoning about knowledge MIT Press". In: *Cambridge, MA, London, England* (1995).

[3] Wojciech Jamroga. "Specification and Verification of Multi-Agent Systems". In: http://krak.ipipan.waw.pl/ wjamroga/papers/jamroga15specifmas.pdf: Institute of Computer Science Polish Academy of Sciences, 2015. ISBN: 978-83-63159-25-2. URL: http://krak.ipipan.waw.pl/~wjamroga/papers/jamroga15specifmas.pdf.

[4] Orna Grumberg, EM Clarke, and DA Peled. "Model checking". In: *International Conference on Foundations of Software Technology and Theoretical Computer Science; Springer: Berlin/Heidelberg, Germany*. 1999.

[5] Massimo Benerecetti. *Appunti del corso di Automated Software Verification*. Slides. 2018. URL: wpage.unina.it/benerece/TSV/Slides-2018/06-CTL-and-Model_Checking.pdf.

[6] Valentin Goranko and Antje Rumberg. "Temporal Logic". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Summer 2024. Metaphysics Research Lab, Stanford University, 2024.

[7] Alfred Tarski. "A lattice-theoretical fixpoint theorem and its applications." In: (1955).

[8] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge university press, 2004.

[9] Rosalie Iemhoff. "Intuitionism in the Philosophy of Mathematics". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2020. Metaphysics Research Lab, Stanford University, 2020.

[10] Luitzen Egbert Jan Brouwer. *Intuitionism and formalism*. 1912, pp. 11–101.

[11] Stefan Bauer-Mangelberg, Jean van Heijenoort, and Stefan Bauer-Mengelberg. "On the Significance of the Principle of Excluded Middle in Mathematics, Especially in Function Theory". In: (1970), pp. 334–345.

[12] Johan van Benthem, Maricarmen Martinez, David Israel, and John Perry. "The stories of logic and information". In: *Handbook of the Philosophy of Information, Elsevier Science Publishers, Amsterdam* (2008), pp. 217–280.

[13] Davide Catta, Vadim Malvone, and Aniello Murano. "Reasoning about Intuitionistic Computation Tree Logic". In: *arXiv preprint arXiv:2310.02355* (2023).

[14] Angelo Ferrando and Vadim Malvone. "VITAMIN: A Compositional Framework for Model Checking of Multi-Agent Systems". In: *arXiv preprint arXiv:2403.02170* (2024).

[15] Joan Moschovakis. "Intuitionistic Logic". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Summer 2023. Metaphysics Research Lab, Stanford University, 2023.

[16] Paolo Mancosu. "From Brouwer to Hilbert: The debate on the foundations of mathematics in the 1920s". In: (1997).

[17] Douglas Bridges, Erik Palmgren, and Hajime Ishihara. "Constructive Mathematics". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Fall 2022. Metaphysics Research Lab, Stanford University, 2022.

[18] Gerhard Gentzen. "Untersuchungen über das logische schließen. I." In: *Mathematische zeitschrift* 35 (1935), pp. 405–431.

[19] Stephen Cole Kleene. "On the interpretation of intuitionistic number theory". In: *The journal of symbolic logic* 10.4 (1945), pp. 109–124.

[20] Stephen Cole Kleene. "Introduction to metamathematics". In: (1952).

[21] Kurt Godel. "Eine Interpretation des Intuitionistis-chen Aussagenkalkuls". In: *Ergebnisse eines mathematisches Kolloquiums* 4 (1933), pp. 39–40.

[22] Kurt Gödel. "Zur intuitionistischen arithmetik und zahlentheorie". In: *Ergebnisse eines mathematischen Kolloquiums* 4.1933 (1933), pp. 34–38.

[23] Stephen Cole Kleene. "EW Beth. Semantic construction of intuitionistic logic. Mededelingen der Koninklijke Nederlandse Akademie van Wetenschappen, Afd. Letterkunde, ns vol. 19 no. 11 (1956), pp. 357–388." In: *The Journal of Symbolic Logic* 22.4 (1957), pp. 363–365.

[24] Helena Rasiowa. "The mathematics of metamathematics". In: (1963).

[25] Saul A Kripke. "Semantical analysis of intuitionistic logic I". In: *Studies in Logic and the Foundations of Mathematics*. Vol. 40. Elsevier, 1965, pp. 92–130.

[26] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism*. Elsevier, 2006.

[27] Per Martin-Lof. *Intuitionistic type theory*. Vol. 6. Bibliopolis Naples, 1984.

[28] Mark van Atten. "The Development of Intuitionistic Logic". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Fall 2023. Metaphysics Research Lab, Stanford University, 2023.

[29] Valery Glivenko. "Sur quelques points de la logique de M. Brouwer". In: *Bulletins de la classe des sciences* 15.5 (1929), pp. 183–188.

[30] Gordon Plotkin and Colin Stirling. "A framework for intuitionistic modal logics". In: *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge (TARK)*. 1986, pp. 399–406.

[31] Alex K Simpson. "The proof theory and semantics of intuitionistic modal logic". In: (1994).

[32] Anupam Das and Sonia Marin. "Brouwer meets kripke: constructivising modal logic". In: *The Proof Theory Blog* (2022). URL: https://prooftheory.blog/2022/08/19/brouwer-meets-kripke-constructivising-modal-logic/.

[33] Frederic B. Fitch. "Intuitionistic Modal Logic with Quantifiers". In: *Portugaliae Mathematica* 7.2 (1948), pp. 113–118.

[34] Gordon Plotkin and Colin Stirling. "A framework for intuitionistic modal logics". In: *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge (TARK)*. 1986, pp. 399–406.

[35] Gisèle Fischer Servi. "Axiomatizations for some Intuitionistic Modal Logics". In: *Rendiconti del Seminario Matematico dell' Università Politecnica di Torino* 42.3 (1984), pp. 179–194.

[36] W. B. Ewald. "Intuitionistic tense and modal logic". In: *Journal of Symbolic Logic* 51.1 (1986), pp. 166–179. DOI: 10.2307/2273953.

[37] M Makkai and G.E Reyes. "Completeness results for intuitionistic and modal logic in a categorical setting". In: *Annals of Pure and Applied Logic* 72.1 (1995), pp. 25–101. ISSN: 0168-0072. DOI: https://doi.org/10.1016/0168-0072(93)00085-4. URL: https://www.sciencedirect.com/science/article/pii/0168007293000854.

[38] Marianna Girlando et al. "Intuitionistic S4 is decidable". In: *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 2023, pp. 1–13.

[39] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[40] Nicolas Markey. "Temporal logic with past is exponentially more succinct". In: *Bulletin-European Association for Theoretical Computer Science* 79 (2003), pp. 122–128.

[41] Philippe Balbiani, Joseph Boudou, Martín Diéguez, and David Fernández-Duque. "Intuitionistic linear temporal logics". In: *ACM Transactions on Computational Logic (TOCL)* 21.2 (2019), pp. 1–32.

[42] Paul Gochet and Pascal Gribomont. "Epistemic logic". In: *Handbook of the History of Logic*. Vol. 7. Elsevier, 2006, pp. 99–195.

[43] Hans Van Ditmarsch, Wiebe van Der Hoek, and Barteld Kooi. *Dynamic epistemic logic*. Vol. 337. Springer Science & Business Media, 2007.

[44] Jerry Seligman and Lawrence S Moss. "Situation theory". In: *Handbook of logic and language*. Elsevier, 1997, pp. 239–309.

[45] D. van Dalen and A. Troelstra. *Constructivism in Mathematics*. Amsterdam: North-Holland, 1988.

[46] Michael Dummett. *Elements of intuitionism*. Vol. 39. Oxford University Press, 2000.

[47] Johan Van Benthem. "Reflections on epistemic logic". In: *Logique et Analyse* 34.133/134 (1991), pp. 5–14.

[48] Johan Van Benthem. *Exploring logical dynamics*. Center for the Study of Language and Information, 1997.

[49] Johan van Benthem. "Semantic parallels in natural language and computation". In: *Studies in Logic and the Foundations of Mathematics*. Vol. 129. Elsevier, 1989, pp. 331–375.

[50] Joseph Boudou, Martín Diéguez, David Fernández-Duque, and Fabián Romero. "Axiomatic systems and topological semantics for intuitionistic temporal logic". In: *European Conference on Logics in Artificial Intelligence*. Springer. 2019, pp. 763–777.

[51] Martín Diéguez and David Fernández-Duque. "An intuitionistic axiomatization ofeventually'". In: *arXiv preprint arXiv:1804.03217* (2018).

[52] ORNA KUPFERMAN and ORNA GRUMBERG. "Buy One, Get One Free!!!" In: *Journal of Logic and Computation* 6.4 (Aug. 1996), pp. 523–539. ISSN: 0955-792X. DOI: 10.1093/logcom/6.4.523. eprint: https://academic.oup.com/logcom/article-pdf/6/4/523/3192763/6-4-523.pdf. URL: https://doi.org/10.1093/logcom/6.4.523.

[53] Rajeev Alur, Thomas A Henzinger, and Orna Kupferman. "Alternating-time temporal logic". In: *Journal of the ACM (JACM)* 49.5 (2002), pp. 672–713.

[54] Wiebe Van der Hoek and Michael Wooldridge. "Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications". In: *Studia logica* 75 (2003), pp. 125–157.

[55] Krishnendu Chatterjee, Thomas A Henzinger, and Nir Piterman. "Strategy logic". In: *Information and Computation* 208.6 (2010), pp. 677–693.

[56] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y Vardi. "A decidable fragment of strategy logic". In: *arXiv preprint arXiv:1202.1309* (2012).

Appendix A

Di seguito, è fornita una lista esaustiva di tutte le figure, tabelle, esempi, definizioni e teoremi presenti nel testo principale. Questo elenco è stato creato per offrire ai lettori un riferimento rapido e conveniente per accedere ai vari elementi che arricchiscono la comprensione del materiale trattato.

List of Figures

List of Tables

List of Example

List of Observation

List of Definition

List of theorems

Ringraziamenti

Desidero esprimere la mia sincera gratitudine a tutte le persone che hanno reso possibile la realizzazione di questa tesi.

In primo luogo, un sentito grazie ai miei relatori, il Prof. **Nello Murano** e il Dott. **Davide Catta**, per la vostra guida illuminata, il vostro costante supporto e la vostra infinita pazienza durante tutto il percorso di ricerca. Senza i vostri preziosi consigli e il vostro incoraggiamento, questo lavoro non avrebbe potuto vedere la luce.

Un ringraziamento speciale va al personale di **ASTREA** per l'inclusione calorosa e il sostegno che mi avete offerto. Grazie a voi ho avuto l'opportunità di partecipare a conferenze stimolanti che hanno arricchito il mio percorso formativo. I pranzi e le cene condivisi con voi sono stati momenti indimenticabili di convivialità e crescita personale. Spero vivamente di poter organizzare presto altre occasioni come queste insieme a voi, per continuare a condividere esperienze e conoscenze.

Un ringraziamento speciale va ai miei **genitori**, per il loro amore incondizionato e il loro sostegno morale in ogni fase del mio percorso accademico. Senza la loro comprensione e il loro incoraggiamento, questo traguardo non sarebbe stato possibile.

Un ringraziamento speciale va a mia sorella, **Francesca**. Grazie per il tuo supporto pratico e per essere sempre presente quando ne avevo bisogno.

Un ringraziamento speciale va ai miei compagni di studio, **Carlo e Gianluca**. Da non ricordo nemmeno quanti anni siamo sempre insieme ad affrontare le avversità accademiche. Il vostro sostegno, la vostra compagnia e la vostra amicizia sono stati fondamentali per superare ogni sfida. Grazie per aver reso questo percorso più gestibile e, soprattutto, più piacevole.

Un ringraziamento speciale va a **Ciampa** per le sue perle di saggezza e le profonde riflessioni a cui mi ha sottoposto. La tua amicizia fraterna e il tuo costante supporto hanno avuto un impatto significativo su di me durante tutto questo percorso. Grazie di cuore per essere stato un amico prezioso e una fonte inesauribile di ispirazione.

Un ringraziamento speciale va alle **mie colleghe** del tirocinio per il loro supporto e la loro collaborazione. In particolare, desidero ringraziare **Federica**, che con la sua preziosa amicizia e il suo spirito positivo, nonché giornate intere a inciuciare mentre facevamo noiosissimi e interminabili test, ha reso l'intera esperienza piacevole e terapeutica. La tua presenza e il tuo aiuto sono stati inestimabili. Grazie di cuore a tutte voi.

Desidero esprimere il mio profondo ringraziamento a **Carla**, **Giggia** e **Maria Elena**, le mie migliori amiche. Siete state non solo compagne di vita, ma anche di avventure, serate memorabili e momenti indimenticabili. Con voi ho condiviso gioie e dolori, risate senza fine e confidenze preziose che hanno reso ogni esperienza ancora più speciale. La vostra presenza costante e il vostro sostegno incondizionato hanno significato tutto per me. Grazie di cuore per avermi arricchito la vita con la vostra amicizia sincera e per essere state sempre lì, pronte a condividere ogni passo del cammino insieme. Non posso che ringraziarvi anche per avermi integrato come uno di famiglia e per avermi permesso di fare altrettanto con voi. La vostra capacità di accogliermi mi ha fatto sentire davvero parte di qualcosa di speciale, e sono grato di poter reciprocamente contribuire alla nostra amicizia.

Desidero esprimere un sentito ringraziamento al gruppo **birretta accademica**. Nonostante non conoscessi tutti voi molto bene all'inizio, mi avete accolto come un fratello. Le serate passate insieme, tra chiacchiere accademiche, birre condivise e quella versione streveza di taboo di cui non ricordo mai il nome, sono state non solo divertenti ma anche fondamentali per rendere mio percorso universitario più leggero.

Desidero ringraziare di cuore il gruppo Aula studio **Loredana**, **Davide**, **Roberto**, **Alessia** e **Anna**. Nonostante gli anni che ci hanno separato, siamo ancora qui, uniti dalla nostra preziosa amicizia che abbiamo coltivato insieme. Le nostre sessioni di studio sono state non solo occasioni per approfondire le nostre conoscenze, ma anche per condividere risate, consigli, sostegno reciproco, partitoni a scopa e bevute. È stato meraviglioso poter contare su di voi durante questo percorso accademico. Grazie per esserci sempre stati e per aver reso ogni momento condiviso così speciale.

Desidero ringraziare di cuore i miei compagni di studio **Roberto** e **Mario**, con cui ho formato un team perfetto e fondamentale per superare alcuni esami. La vostra competenza, dedizione e supporto reciproco sono stati determinanti nel raggiungere i nostri obiettivi accademici. È stato un onore studiare e crescere insieme a voi. Grazie per aver reso questa esperienza educativa così significativa e gratificante.

Desidero esprimere il mio sincero ringraziamento **Gruppoon** per le meravigliose uscite insieme, le chiacchiere che hanno reso ogni momento speciale, le buonissime braciate che mi hanno unito a voi ancora di più. È stato un privilegio essere accolto da voi come un fratello. In particolare, voglio ringraziare **Federica** per il suo supporto fondamentale: i nostri lunghi viaggi in treno, le lunghe chiacchierate, nonostante fossero di prima mattina

e con la puzza di merda nel treno. Grazie per la tua compagnia preziosa e per gli sfoghi condivisi che sono stati dei veri punti di riferimento per me. Sono profondamente grato per ogni momento vissuto insieme e per il vostro costante sostegno.

Desidero ringraziare di cuore i miei cari amici di una vita: **Mario**, **Paolo**, **Vincenzo**, **Petronz**, **Daniele**, **Adriana**, **Maurizio** e **Falko**. Con voi ho condiviso risate, momenti di riflessione e tanti bei ricordi. La vostra amicizia è stata un faro costante nella mia vita, rendendo ogni giorno più luminoso e significativo. Grazie per essere sempre stati al mio fianco, per il vostro sostegno sincero e per i momenti speciali che abbiamo condiviso insieme. Siete davvero parte integrante della mia storia e del mio cuore.

Vorrei ringraziare di cuore **Giuliana** e **Alessandra**, due amiche straordinarie che hanno arricchito la mia vita in modi unici. Oltre ad essere compagne di vita quotidiana, siete state le mie compagne di concerti e festival. Grazie per le indimenticabili serate trascorse insieme ad ascoltare musica dal vivo, per i momenti di entusiasmo condiviso durante i festival e per le esperienze che abbiamo condiviso nel cuore della musica. La vostra amicizia è un tesoro prezioso che custodirò per sempre nel mio cuore. Grazie per essere state sempre lì, rendendo ogni occasione speciale e memorabile.

Desidero esprimere il mio profondo ringraziamento ad **Agritech Academy**, in particolare a **Marianna**, **Mario**, **Faber**, **Ivana** , ma soprattutto ad **Alessio**, il mio compagno di follie e collega in progetti ambiziosi. Grazie per essere sempre pronto a correre avventure insieme e per il tuo sostegno instancabile. Le nostre idee pazze e le risate condivise hanno reso ogni giornata di lavoro e non un'esperienza divertente, gratificante e unica. Non c'è mai stato un momento noioso con te! Sono grato di avere un collega così entusiasta e positivo. Guardo avanti a molte altre avventure, bevute e postegge insieme a te.

Vorrei esprimere un sentito ringraziamento al mio gruppo di videogames, **Alessia**, **Riccardo** e **Attilio**, il mio salvatore personale. Grazie per essere stati una fonte di divertimento, supporto e amicizia e di chiacchierate notturne. In particolare, un immenso grazie va a Attilio, che oltre alle ore di gioco trascorse, è stato anche il mio eroe personale per aver riparato il mio PC in un momento critico. Senza il tuo intervento tempestivo e competente, non avrei potuto completare il mio percorso di laurea. La tua generosità e la tua expertise sono state fondamentali e non potrò mai ringraziarti abbastanza. Siete stati più che compagni di gioco, siete stati veri amici. Guardo avanti a molti altri momenti epici insieme!

Vorrei dedicare un enorme ringraziamento all'equipe di **animazione** con cui ho avuto il privilegio di lavorare. In quei due mesi, siete stati più di colleghi: siete diventati la mia famiglia lontano da casa. Grazie per avermi spinto a superare i miei limiti e per avermi aiutato a crescere personalmente e professionalmente. Ogni ora trascorsa nelle prove, è stata un'occasione per imparare e migliorare insieme. Non dimenticherò mai le risate davanti agli sketch, le sparlate sugli ospiti, i cornetti del Makako alle 2 di notte, e persino la pasta al sugo immangiabile che ci propinavano ogni santo giorno. Ognuno

di voi ha lasciato un'impronta indelebile nel mio cuore. Grazie di cuore per l'esperienza indimenticabile e per avermi dato tanto da ricordare.

To my dear friend **Oksana**. I wanted to take a moment to express my gratitude and appreciation for your friendship, even though we are far apart. Distance may separate us physically, but the bond we share transcends any geographical boundaries. Your warmth, kindness, and unwavering support have been a constant source of strength for me. Whether it's sharing stories, offering advice, or simply being there for each other, I cherish every moment we've spent together. Thank you for being a part of my life and for the countless memories we've created. I look forward to the day when we can meet in person. Until then, know that you are always in my thoughts and in my heart.