Per progetto: se rete è free choice e è sound allora è safe. Se è safe è bounded

Rete debolmente connessa: Quando esiste un percorso non direzionato tra ogni coppia di nodi distinti la rete si dirà **debolmente connessa**. Una rete è debolmente connessa se e solo se non è possibile dividerla in componenti separate.

Un **percorso non direzionato** (undirected) è una sequenza non vuota di nodi che soddisfa per ogni .

Rete fortemente connessa: Quando esiste un percorso tra ogni coppia di nodi distinti la rete si dirà **fortemente** **connessa**. Una rete è fortemente connessa se e solo se per ogni arco esiste un percorso da a .

S-invariant: di una rete N è una soluzione razionale dell’equazione x\*N=0, dove x è un vettore che ha tante posizione quanti il numero di places della rete e N è una matrice di incidenza.

In maniera alternativa, ogni transizione nella rete produce tante risorse quante ne consuma

Proprietà fondamentale degli S-invariant: Sia I un invariante di N e M appartenga all’ insieme di markings raggiungibile da M0, allora I\*M = I\*M0

Prova:

Sia , esiste tale che . Dal Lemma dell’equazione del marking si sa che

Quindi:

Invariante può essere Positivo, tutti i suoi valori sono positivi, oppure semi-positivo, tutti i suoi valori sono non negativi e almeno uno è positivo

Condizione sufficiente per la proprietà di boundedness: Se (P, T, F, M0) ha un S-invariant positivo, allora la rete è bounded. Questo significa che qualunque sia il marking iniziale, la rete sarà bounded.

Condizione necessaria per la proprietà di liveness: Se (P, T, F, M0) è live, allora per ogni invariante semi-positivo I, si avrà che I\*M0 > 0.

Il vantaggio di questo teorema sta nel fatto che, se si calcola l’invariante del sistema e si trova un invariante semi-positivo tale che I\*M0 = 0, allora si può concludere che il sistema non è live.

T-Invariant: di una rete N è la soluzione razionale all’ equazione y\*N=0, dove y è un vettore che ha tante posizioni quante sono le transizioni e N è una matrice di incidenza.

Proprietà fondamentale dei T-invariant: Sia M’ abilitato da M tramite la funzione di transizione t, il vettore di Parikh è un T-invariant se e solo se M’ = M

Boundedness, Liveness, T-invariant positivo: se un Sistema è live e bounded, allora esso ha un T-invariant positivo.

Strong Connectedness: se un sistema debolmente connesso è live e bounded allora è strong connected

Strong Connectedness tramite invariant: se un Sistema debolmente connesso ha un S-Invariant positivo e un T-Invariant positivo, allora è fortemente connesso

Workflow net Sound: se presenta le seguenti 3 caratteristiche: no dead task, option to complete e proper completion.

No dead task, nessuna transizione deve essere dead (una transizione è dead quando non si può raggiungere nessun altro marking da tale transizione).

Proper completion, se il place finale risulta marked, nella workflow net, non ci deve essere nessun altro place con un token all’interno.

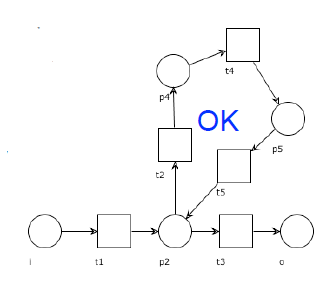
Option to complete, prima o poi il place finale sarà marked.

Per verificare la soundness si può applicare un’analisi bruteforce. Prima si verifica che la rete sia una workflow e poi si costruisce un grafo di raggiungibilità in cui si verifica che:

-per ogni transizione, ci deve essere un arco con etichetta relativa alla transizione

-ci deve essere un unico place iniziale.

-Il place finale sia raggiungibile da ogni place nel grafico e contenga un solo token



Tale analisi però è molto costosa, quindi si usa un altro approccio basato su workflow net sound, liveness e boundedness, grazie a una rete N\* in cui si inserisce una transizione detta reset che permette di tornare al place iniziale dal place finale

Soundness, liveness, boundedness: Sia N = (P, T, F) una workflow net e sia reset non appartenente all’insieme delle sue transizioni. Se N è sound, allora N\*, in cui reset appartiene all’insieme delle sue transizioni è live e bounded

Prova:

-N\* live e bounded implica N sound:

Poichè N è live, per ogni transizione esiste un marking appartenente all’insieme dei marking raggiungibili da i, place iniziale, che la abilita, quindi vale anche per reset

Consideriamo adesso qualche marking M che abilita reset, questo significa che M contiene almeno un token nel place finale O.

Il firing di reset, rimuove un token in O e lo produce in I, ovvero O+L = M che abilita M’ tramite la funzione reset, dove M’ = I+L, allora M’ appartiene all’insieme dei marking raggiungibili da I e M’ contiene I.

Poiche N è bounded, l’unica possibilità per M’ è che L = vuoto e quindi che M = O, altrimenti tutti i place marked da M’ – I = M -O sarebbero undounded

Questo significa che ogni volta che si abilita reset, è presente un solo token in O. N\* quindi garantisce Option to complete e proper completion, mentre no dead task è garantita per assunzione

-N sound implica N\* live e bounded: per dimostrare questo è necessario prima chiarire il seguente lemma.

Se N è sound, un marking M è raggiungibile in N se e solo se M è raggiungibile in N\*

L’implicazione diretta è banale, per l’implicazione inversa si consideri i che abilita M, tramite la sequenza di transizioni sigma in N\* e si dimostri per induzione sul numero di istanze della transizione reset che appare nella sequenza sigma.

Per il caso base si ha quando il numero di istanze r = 0, ovvero quando la funzione reset non viene completamente usata in sigma. Quindi, tutte le transizioni che appaiono in sigma sono transizioni appartenente alla rete N e M è raggiungibile in N.

Per il caso in cui r>0, si consideri k come il primo indice in cui la funzione reset appare in sigma. Di conseguenza sigma = sigma’ tk sigma ‘’, con sigma’ che può contenere altre occorrenze di reset.

A questo punto si sa che sigma’’ contiene soltanto r-1 istanze di reset ed è più piccolo di sigma. Quindi per induzione si avrà che M è anche raggiungibile in N

A questo punto:

per provare la boundedness si procede per contraddizione assumendo che N\* è unbounded.

Poiché è unbounded, allora con . Sia l’insieme di place in più che si trovano in . Poiché è sound per assunzione, allora e, poiché è più grande di , è possibile rieseguire la computazione a partire da e raggiungere un marking (dal Lemma di monotonicità). Una delle condizioni di soundness, in questo modo, è stata violata perché non è possibile che esista un marking che contenga, insieme ad , altri token in altri place; di conseguenza, la rete non può essere sound se non è bounded.

Per provare la liveness di , si prenda una qualsiasi transizione (possibilmente ‘reset’) e si consideri un marking raggiungibile in . Dal Lemma tecnico per induzione, è noto che è raggiungibile anche in . Poiché è sound per assunzione, allora (‘Option to complete’ e ‘Proper completion’) e (‘No dead task’). Si consideri, adesso, , allora in e .

Dal Teorema di Strong Connectedness si sa che una rete che non è, allo stesso tempo, live e bounded non è fortemente connessa. Quindi, la prima cosa da verificare, in questo contesto, è che sia fortemente connessa, proprio perché, altrimenti, non può essere live e bounde.

-Building blocks: tecnica usata per garantire che la workflow sia sound per costruzione.

Quello che si fa è selezionare un set di building blocks, ovvero un insieme di piccolo workflow nets per cui sia facile dimostrare che sound e safe tramite approccio bruteforce.

Dopo si fa una composizione di tali blocks sound e safe di modo da poter ottenere altre workflow net sound e safe.

Il building block più semplice contiene un input place una transizione e un output place. Dopo per semplicità abbiamo una sequence in cui due transizioni sono poste in sequenza. Altri building blocks sono implicit xor e iteration, Per costruire una workflow net sound e safe possono essere usate strategie top down di refinement (partire da un attività atomica e aggiungere attività piu complesse poco alla volta) o bottom up di abstraction (partire da un’attività complessa e sostituire eventuali building blocks conosciuti con singole transazioni)

Sapendo che una rete è s-net possiamo dire qualcosa riguardo la sound

Una rete è **S-net** se . Un sistema è un **S-system** se è un S-net.

Se una workflow net è un S-system, allora essa è sound, ma anche safe.

Prova:

Sapendo che è un S-system, anche è un S-system. Se e sono S-systems, allora e sono bounded e safe. Per assunzione, si sa che è una workflow net, quindi e fortemente connessa. Inoltre, si sa che e questo, insieme al fatto che è fortemente connessa, implica che è live. Poiché è live e bounded, allora è sound.

S-Net: una rete è S-net se ogni transizione ha un single input e un single output, per ogni transizione che faccia parte del set delle transizioni della rete. Un sistema(N,M0) è un s-system se N è un s-net

Una workflow net N è una S-Net se e solo se N\* è un S-net

Proprietà fondamentale degli S-Systems: Sia (P, T, F, M0) un S-System. Se M è un marking raggiungibile, allora M0(P) = M(P), dove P è il set di place di N.

Corollario: qualunque S-Systems è bounded

S-invariants delle S-Nets: Sia N = (P, T, F) una S-net connessa. Un vettore I: S -> Q è un S-invariant di N se I ha lo stesso peso per ogni place, per un qualche valore k

Liveness per gli S-Systems: Un S-System è live se e solo se N è fortemente connesso e M0 marca almeno un place

Prova:

- S-system live implica N fortemente connessa e M0 marca almeno un place:

Per assunzione il sistema è live, ma anche bounded, perché è un S-system. Dal Teorema di strong connectedness è noto che è fortemente connessa. Poiché è live, il place di qualunque delle sue transizioni è marked, quindi . Questo significa che con .

- N fortemente connessa e M0 marca almeno un place implica S-System live:

Dato qualunque marking raggiungibile e qualunque transizione , si vuole mostrare che è possibile raggiungere un altro marking .

Si consideri un place , che esiste perché per ipotesi e . Poiché, per assunzione, la rete è fortemente connessa, è noto che esiste un percorso da ogni nodo della rete ad ogni altro nodo; quindi, in particolare, esiste un percorso da al pre-set di , ovvero esiste un percorso . Dalla definizione di S-system si sa che

quindi: per

Cosa si può dire di una wf net che è S-system ?

Se un WF N è un S-system, allora esso è sound e safe.

Prova:

Sapendo che è un S-system, anche è un S-system. Se e sono S-systems, allora e sono bounded e safe. Per assunzione, si sa che è una workflow net, quindi e fortemente connessa. Inoltre, si sa che e questo, insieme al fatto che è fortemente connessa, implica che è live. Poiché è live e bounded, allora è sound.

T-Net: una rete è T-Net se ogni place ha un single transiction input e un single transition output. Un sistema (N,M0) è T-System se N è un T-net

Liveness per T-Systems: Un T-System (N,M0) è live se e solo se ogni circuito N è inizialmente marked M0.

Rete free-choice: se ogni volta che esiste un arco (p,t), allora vi è un arco da un input place di t a una transizione di output di p.

Proprietà fondamentale delle free-choice nets: Sia una free-choice net. Se e , allora .

N\* di free-choice: Una workflow net N è free-choice se e solo se N\* è free-choice.

Teorema di Rank: un sistema free-choice è live e bounded se:

-ha almeno una transizione e un place

-è connesso

-Mo marca ogni proper siphon

-ha un S-invariant positivo

-ha un T-invariant positivo

-rank(N) = |Cn| - 1, dove Cn è il set di cluster meno 1, rank(n) è il numero di righe e colonne linearmente indipendenti della matrice di incidenza

Cluster di X: sia X un nodo nella rete N. il Cluster di X è il più piccolo set tale che:

- x appartiene al cluster di X

- se p appartiene al cluster di X intersecato P (insieme di place della net), allora tutte le transizioni nel post set di p sono nel cluster

- se t appartiene al cluster di X intersecato T (insieme di transizioni della net), allora tutti i place nel preset di t appartengono al cluster.

Come si Trova un cluster: si parte dal place, si inseriscono tutte le transizioni nel post set di P, si inseriscono tutti i place nel pre set delle transizioni.

Siphon: un set di place R è un siphon se il preset di R è contenuto nel post set di R, cioè se tutte le transizioni che possono produrre token nel set di place richiedono qualche token che è gia nel set. In particolare, esso è un proper siphon se R != vuoto.

Proprietà siphon: i siphon non marked rimangono unmarked. Se un sistema ha un proper siphon unmarked, allora il sistema non è live.

Corollario alla prop. fondamentale siphon: Se un siphon R è marked in qualche marking raggiungibile M, allora era inizialmente marked in M0

Siphons e liveness: se un sistema è live, allora ogni proper siphon R è marked

S-component: Sia N = (P, T, F) una rete e prendi un set di nodi di tale rete e sia N’ una sotto rete di N. La sottorete N’ è S-component se:

- è una S-net fortemente connessa

-Se si prende un place, tutte le transizioni connesse ad esso devono essere selezionate

S-Coverability: se un free-choice system è live e bounded, allora esso è S-Coverable. Di conseguenza se la rete è free-choice ma non S-Ceverable allora è non live e non bounded.

EPC: ogni diagramma deve avere un evento di start e di fine e sono elementi detti passivi, utilizzati per descrivere una circostanza o condizione e non corrispondono alle attività. Le funzioni sono unità di lavoro e rappresentano elementi attivi del diagramma. I connettori logici sono utilizzati per descrivere il modo in cui il flusso viene diviso o unito tra due branch del workflow. Eventi, funzioni e connettori sono connessi da frecce tratteggiate.

BPMN: definisce uno standard per i diagrammi dei processi aziendali, basato su una tecnica di flow-charting, i suoi oggetti grafici si raggruppano in quattro categorie principali:

-flow object: come cerchi per gli eventi, rombi per i connettori e rettangoli per le attività

-connectinc object: come le frecce per I flussi di sequenze e nuove frecce per lo scambio di messaggi o commenti.

-artefacts

-swimlanes: utilizzati per assegnare responsabilità all’esecuzione dell’attività

BPMN vs EPC: Preferisco BPMN perche permette di rappresentare bene la coreography. Per l’Epc non ci sono regole precise su come rappresentare i diagrammi, mentre per BPMN si.

BPMN è una notazione semplice, i simboli sono estensibili, nel senso che aggiungendo piccole decorazioni è possibile cambiare e rendere più dettagliato il significato di ogni simbolo, è intuitivo, permette di gestirte sia diagrammi di orchestartion che di coreography.

Tradurre BPMN in Petrinet:

dobbiamo fare un assunzione: ogni gateway che ha piu input o piu output deve essere trasformato in due gateways collegati. In maniera approssimativa eventi e attività vengono tradotti in transizioni, frecce vengono tradotte in places.

Gli step per converire il flusso:

-converire i flussi di sequenza e di messaggi in place

-convertire gli oggetti del flusso

-assicuraare un input place e un output place

Tipi di Astrazione:

Nel contesto della gestione dei processi aziendali esistono 3 tipi di astrazione:

orizzontale, di aggregazione e verticale.

Per quanto riguarda quella orizzontale, questa attua una separazione dei diversi modeling levels a cui si deve lavorare.

Si divide in livello M0 in cui troviamo le istanze concrete coinvolte nel processo, M1 in cui si gruppano classi di istanze insieme al fine di catturare la loro struttura, M2 in cui si definiscono i concetti usati per costruire il modello e M3 in cui si definiscono le entità astratte usate per definire i concetti del meta-modello.

Per quanto riguarda l’astrazione di aggregazione, l’idea è di aggregare per unire diverse attività o funzionalità differenti, giustificando un diverso livello di granularità. La differenza co quella orizzontale è che qui ad ogni livello, tutte le entità rimangono allo stesso livello di granularità.

Infine in quella verticale, si ha una distinzione dei vari modelli coinvolti nella descrizione del processo aziendale. Si è interessati alla funzione associata al processo, con le informazioni manipolate dal processo con l’organizzazione in cui il processo viene avviato e sulle tecnologie utilizzate per realizzarlo.

Ciclo di vita di un business process:

Un modello lifecycle è la descrizione dei passaggi coinvolti nello sviluppo e costruzione di un software o prodotto. Il ciclo di vita dei processi aziendali è simile all’approccio PDCA (Plan, Do, Check, Act), si pianifica le azioni da eseguire, si eseguono, si verifica che gli obbiettivi prefissati siano stati raggiunti, si agisce per cambiare il piano d’azione e migliorare la performance del processo. Le varie fasi del ciclo di vita di un BP sono: Design e Analysis in cui si identifica e modella il processo aziendale e in seguito si esegue una validazione tramite strumenti come workshop in cui più persone controllano che tutte le istanze del processo siano riflesse dal modello, simulazione a supporto della validazione e verifica che consente nell’analizzare e migliorare il modello del processo aziendale per assicurarsi che alcune proprietà vengano rispettate. Dopo abbiamo la Configuration in cui si sceglie il sistema e si sceglie il modo in cui testare e distribuire il modello al fine di implementarlo. Poi abbiamo Enactment che si sostanzia nell’esecuzione del processo, nel monitoraggio e mantenimento di questo. Infine, Evaluation, cioè la valutazione delle informazioni monitorate. Al centro del modello del ciclo di vita abbiamo una parte dedicata ad Administration e Stakeholders, utile per ricordarsi che tutti gli artefatti prodotti in questa fase devono essere descritti, organizzati, gestiti in repository.

Altri tipi di modelli che possiamo avere sono il modello a cascata, è un processo sequenziale di design del software, visto come flusso verso il basso che attraversa le varie fasi: requisiti, design, implementazione, verifica e manutenzione.

Oppure il modello di Extreme programming la cui idea è quella di iniziare con un prototipo grezzo, che viene dato all’utente per ricevere feedback e migliorarlo ad ogni iterazione.

Differenza tra business process management e business process management system

BPM: la gestione dei processi aziendali include concept, metodi e tecniche di support al design, all’amministrazione, alla configurazione, alla messa in atto e all’analisi dei processi aziendali.

BPMS: Il **sistema di gestione dei processi aziendali** è un sistema software generico che è guidato da rappresentazioni del processo esplicite per la coordinazione della messa in atto del processo aziendale.

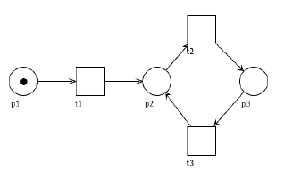
Liveness e Deadlock-freedom: Una rete è live se ogni transizione si può ripetere. Una transizione è live se per ogni marking raggiungibile M, può essere raggiunto un altro marking M’.

Una rete è deadlock-free, se ogni marking raggiungibile abilita qualche transizione. Quindi, in ogni momento della computazione, è garantito che qualche transizione può fare il firing.

L’implicazione è che se una rete è live, allora è deadlock free.

Dimostrazione:

Per contraddizione, sia con e ( non può essere vuoto). Dalla liveness si sa che, ma (per assunzione), quindi si ha che . Questo vuol dire che si dovrebbe avere , ma ciò è assurdo perché l’ipotesi inziale era che .



Cosa sono le sequenze di errore?

Sequenze di firing tali che ogni sua continuazione porterà ad un errore, inoltre è la più breve sequenza possibile per cui l’errore apparirà

Marking equation lemma e matrice di incidenza (punto in comune è il vettore di Parikh)

Il marking equation Lemma afferma che se M -> M’ tramite la sequenza di transazioni sigma, allora M’ = M + N\*vettore di parik.

Caso Base: sigma = vuoto

Passo induttivo = sigma = sigma’ t

Se la sequenza è di questo tipo, allora . L’ipotesi induttiva è che .

Sia N = (P, T, F) unaa rete, ma matrice di incidenza sostanzialmente è una tabella in cui i valori che possono essere inseriti sono -1 se p è nel pre-set di t, +1 se è nel post set, 0 alrimenti.

Monotocinity lemma : Se M -> M’ tramite sequenza di transazioni sigma, allora M+L -> M’+L per ogni L.

Soluzione anche qua cosa caso base sigma = 0 e passo induttivo sigma = sigma’ e transazione t.

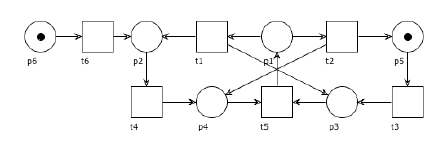
Boundedness lemma: Se un sistema è bounded e con , allora .

Sia e si supponga che . Ciò significa che esiste un marking con : una possibilità è che sia vuoto; quindi, si ponga .

Dal Lemma della monotonicità si avrà che ; questo significa che ogni è raggiungibile da ma, poiché il sistema è bounded, . Se contenesse un place, allora avrebbe almeno token in quel place e, poiché è raggiungibile da ogni , non sarebbe possibile stabilire un bound sul numero di token da mettere nel place. Quindi, se il sistema è bounded, quando si eseguono delle computazioni, non è possibile creare più risorse di quelle che si hanno all’inizio.

Il Lemma di boundedness è molto importante perché se si mostra che un marking è raggiungibile e che include strettamente (), allora si può concludere che il sistema non è bounded, perché vi è la possibilità di creare nuovi token.

Esempio di rete deadlock-free ma non live

La rete non è live perché , ad esempio, non può essere abilitata dopo il primo firing; la rete è deadlock-free e bounded (safe, in particolare), ma non è ciclica.

Definizione formale di live implica place-live e dimostrazione

I sistemi live sono place-live. Partiamo dicendo che Una transizione t è live se, per ogni marking raggiungibile M, può essere raggiunto un altro marking M’ che abilita t.

Un place è live se, ogni volta che diventa unmarked, c’è la possibilità che ridiventi marked.

Dimostrazione:Dato ogni e . Si vuole trovare tale che .

Il sistema è live e consideriamo ogni transizione connessa al place , ovvero ,

che indica come debba appartenere al pre-set o al post-set di .

Dal momento che la rete è live, è possibile raggiungere ed esiste tale che :

* Se è nel pre-set di , allora dopo il firing di si avrà
* Se è nel post-set di , allora dopo il firing di si avrà

Per verificare che un place sia live o dead si può ispezionare il grafo di raggiungibilità:

* Liveness, da ogni nodo del grafo si può raggiungere un nodo, seguendo un percorso, con un token in ;
* Dead, se tutti i nodi del grafo non hanno un token in

Implicazioni di S-invariant positivo

Intanto partiamo dal presupposto che un S-Invariant di una rete N è una soluzione razionale dell’equazione x\*N = 0, in cui X è un vettore che ha tante posizioni quant’è il numero dei place e N è una matrice di adiacenza. Quando si moltiplicano si ottiene sempre il vettore 0.

Un S-invariant è positivo se tutti i valori al suo interno sono positivi. L’implicazione è: se (P,T,F,M0) ha un S-invariant positivo, allora la rete è bounded

Prova:

Sia e sia un S-invariant positivo. Si consideri un singolo place , allora si vuole calcolare il prodotto tra i pesi dei token che si trovano nel place al marking . Questa quantità può essere ottenuta come

La quantità calcolata è indipendente dal marking che si raggiunge, quindi e il peso dei token nel place è minore o uguale a questa costante. Adesso, poiché tutti gli elementi di sono positivi, si può ottenere senza cambiare la relazione di ineguaglianza:

Questo ragionamento può essere esteso e il valore più alto può essere considerato il bound valido per tutta la rete. Il vantaggio di questo teorema sta nel fatto che, se si è in grado di ottenere un S-invariant positivo per la rete, ciò prova che il sistema è bounded per qualsiasi marking iniziale. Inoltre, se si vuole stimare un bound per un particolare place, esso può essere ottenuto semplicemente calcolando

Vettore di parikh e sequenza di transizioni

Sia N = (P,T,F) una rete e sigma appartenente a T\*, una sequenza finita di transizioni. Il vettore di parik indica per ogni transizione t di T, il numero di occorrenze, firinig di t in sigma. Una sequenza di transizioni è una sequenza che puo essere finita o meno e con cui si può raggiungere o meno partendo da un marking iniziale M, il marking finale M’

Spiega cosa sono gli orchestration e i collaborationDal momento che i processi aziendali sono attuati in una singola organizzazione per definizione, l’ordinamento delle attività può essere controllato da un sistema di gestione dei processi aziendali. Questo controllo centralizzato è molto simile a un direttore d’orchestra che controlla i propri musicisti e, per tale motivo, i processi aziendali sono anche chiamati **process orchestrations**. Tali processi forniscono una visione dettagliata delle attività e dei loro vincoli di esecuzione e i modelli basati sugli eventi.

I **processi aziendali collaborativi** invece, sono dei processi il cui scopo è anche quello di tracciare esattamente cosa si sta facendo e quali relazioni causali intercorre tra i task del progetto.

Descrive quindi le interazioni di un set di processi aziendali.

La differenza con l’orchestration è l’assenza di un agente centrale che controlla le attività del processo di business. Un esempio pratico sono le ballerine in una coreografia che agiscono in maniera autonoma.