

Politecnico di Milano



AA 2005 / 2006

**Progetto Alloy : Verifica specifiche sistema informativo
aziendale**

Silvia Barbariga (675460)
Paolo Tagliaferri (675331)

Descrizione del progetto

Le aziende che si occupano di compravendita di materie prime sono divise in 3 settori: il settore che si occupa di ritirare la merce, il settore in cui la merce viene verificata e il settore che si occupa delle spedizioni. I tre settori sono coordinati da dei direttori che non appartengono a nessun settore in modo specifico, ma sono di pari grado tra di loro e superiori a tutti gli altri dipendenti dell'azienda. Ogni singolo settore è composto da tre categorie di lavoratori: i manager, gli impiegati e i fattorini. I manager sono i gestori del settore e sono nell'organizzazione dell'azienda superiori a impiegati e fattorini del loro settore. Gli impiegati, che si occupano di svolgere mansioni standard, sono superiori solo ai fattorini del loro settore. I lavoratori di un'azienda essendo persone, sono soggetti alle normali relazioni tra persone, quali amicizie antipatie, etc.. I sentimenti personali potrebbero influenzare le decisioni lavorative per cui se una persona è amica di un'altra ed è anche il suo superiore, potrebbe avvantaggiarla. Nell'azienda si vogliono evitare situazioni di questo tipo, per cui è stato introdotto un sistema informativo che contiene tutti i dati e le relazioni tra le persone che, qualora si voglia assegnare a una persona un certo ruolo in un determinato settore, risponde se il nuovo ruolo non avvantaggia la persona e non la porta a fare il superiore di amici e no altrimenti. Si verifichi, utilizzando Alloy, che non può succedere che il sistema informativo sbagli la sua risposta.

Modello proposto e assunzioni

Per implementare correttamente il modello sopra descritto è necessario porre alcuni vincoli e definire le assunzioni fatte nel nostro modello del problema.

Secondo le nostre assunzioni:

- Un'azienda è guidata da almeno un dirigente, da dipendenti con mansioni differenti (tra le tre possibili) ed è composta da tre settori, uno per ogni tipologia possibile. In questo modello si contempla la presenza di una sola azienda.
- Tutti i soggetti umani coinvolti sono stati modellizzati a partire da una generica entità INDIVIDUO, estesa e specializzata nelle entità DIPENDENTE e DIRIGENTE. Per queste entità è stato previsto un insieme di attributi minimale per le nostre necessità, ma facilmente

estendibile in caso si voglia ampliare l'ambito di applicabilità di questo progetto.

- La struttura aziendale è formata da tre settori: uno relativo al ritiro della merce, uno alla verifica e uno alla spedizione
- I dirigenti non sono associati esplicitamente ai settori: infatti si assume che l'azienda sia guidata da un team direttivo che coordina le operazioni in ogni settore aziendale.
- Per ogni settore si prevede un numero minimo di tre dipendenti, uno per ogni mansione: FATTORINO, IMPIEGATO, MANAGER. Inoltre si assume che un dipendente possa lavorare esclusivamente nel settore di sua appartenenza e in un'unica mansione.
- Si assume che le gerarchie valgano all'interno di un settore preso singolarmente (tranne che nel caso dei dirigenti, che sono superiori a tutti gli altri). Nello specifico, un fattorino si trova alla base della piramide aziendale, sovrastato da impiegati e dai manager (che sono *de facto* i dirigenti del settore)

Alcune delle assunzioni fatte possono sembrare restrittive e poco corrispondenti alla realtà. Infatti è plausibile pensare, ad esempio, che in un settore la cardinalità dei quadri inferiori sia più elevata rispetto a quella dei quadri manageriali. Inoltre si potrebbe stabilire un numero minimo di dirigenti pari al numero di settori.

Tuttavia non è stato possibile allargare eccessivamente lo scope del nostro modello, implementando quindi caratteristiche più realistiche di quelle sopra mostrate: questo perchè le risorse computazionali richieste (dovute alla crescente complessità e numerosità del modello) crescono esponenzialmente all'aumentare delle entità in gioco e proporzionalmente crescono anche i tempi di processing.

Studiando le assunzioni fatte, abbiamo scoperto che il numero minimo da associare allo scope per rendere sensata la simulazione è pari a 10 individui (3 dipendenti * 3 settori + 1 dirigente). Il numero ottimale utilizzato nelle nostre simulazioni è pari a 12 individui, oltre il quale i tempi richiesti dalla simulazione sono eccessivi.

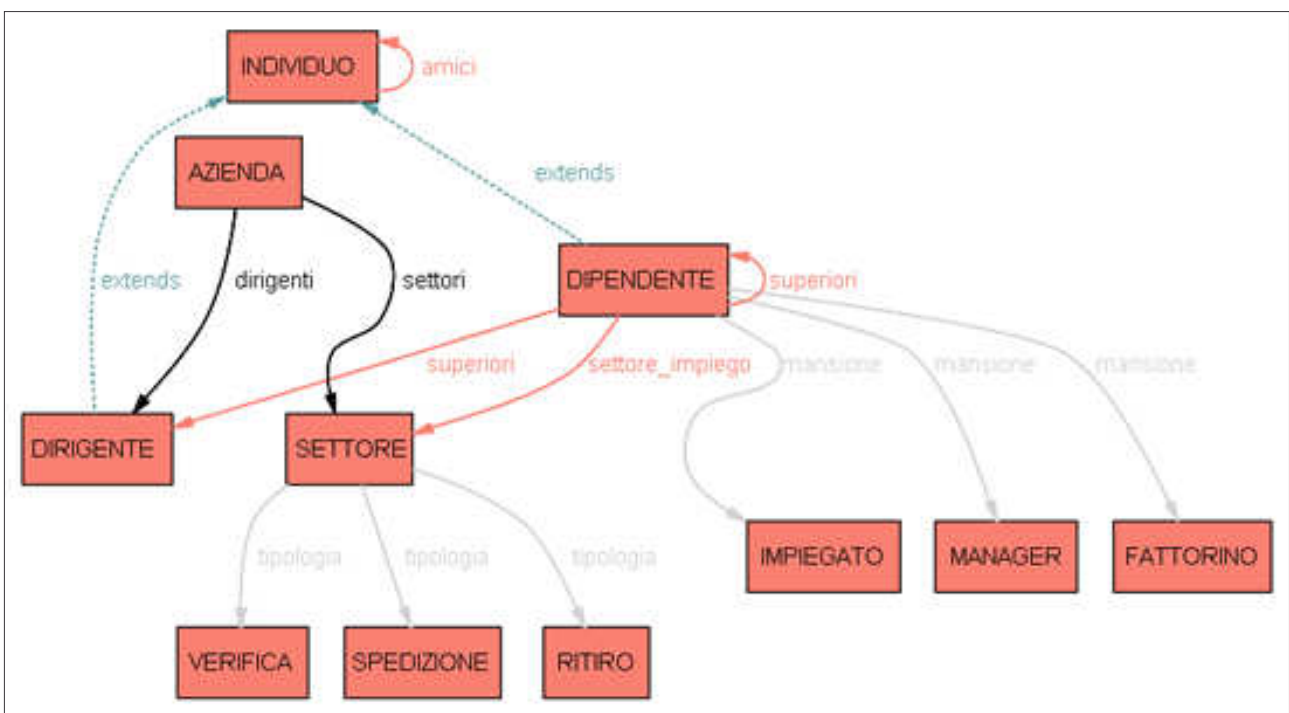
Nella simulazione sono state considerate due relazioni tra gli individui presenti in azienda: **amicizia** e **gerarchia**:

- **Gerarchia**: questa relazione gode della proprietà transitiva, ma non delle proprietà riflessiva e simmetrica
- **Amicizia**: questa relazione gode delle proprietà simmetrica, ma non delle proprietà transitiva e riflessiva

Per implementare queste proprietà si è reso necessario introdurre alcuni fatti:

- Per la relazione di *gerarchia* le proprietà vengono intrinsecamente specificate in singoli fatti relativi ad ogni mansione, poiché si fornisce una regola di comportamento per ogni caso specifico, definendo in modo assoluto gli elementi che appartengono all'insieme dei superiori per ogni dipendente presente nella simulazione
- Per la relazione di *amicizia*, si sono esplicitate solamente le proprietà riflessiva e simmetrica tra due individui generici, in quanto non è possibile asserire nulla riguardo alla proprietà transitiva. Inoltre è stato implementato il vincolo che impedisce l'amicizia tra due individui legati da una relazione di gerarchia.

Riportata qui sotto abbiamo la rappresentazione del metamodello così definito generata da Alloy



In azzurro abbiamo le relazioni di estensione, in rosso le relazioni tra entità, in nero le composizioni e in grigio le specializzazioni.

Model Checking

Le simulazioni effettuate sono state suddivise in due tipi:

- Checking delle assert: abbiamo definito quattro asserzioni per verificare alcune proprietà notevoli del modello:
 1. **DirettoriNoAmici**: i dirigenti non possono avere amici tra i dipendenti
 2. **FattoriniNoAmiciSettore**: i fattorini non possono avere amici tra i manager e gli impiegati del proprio settore

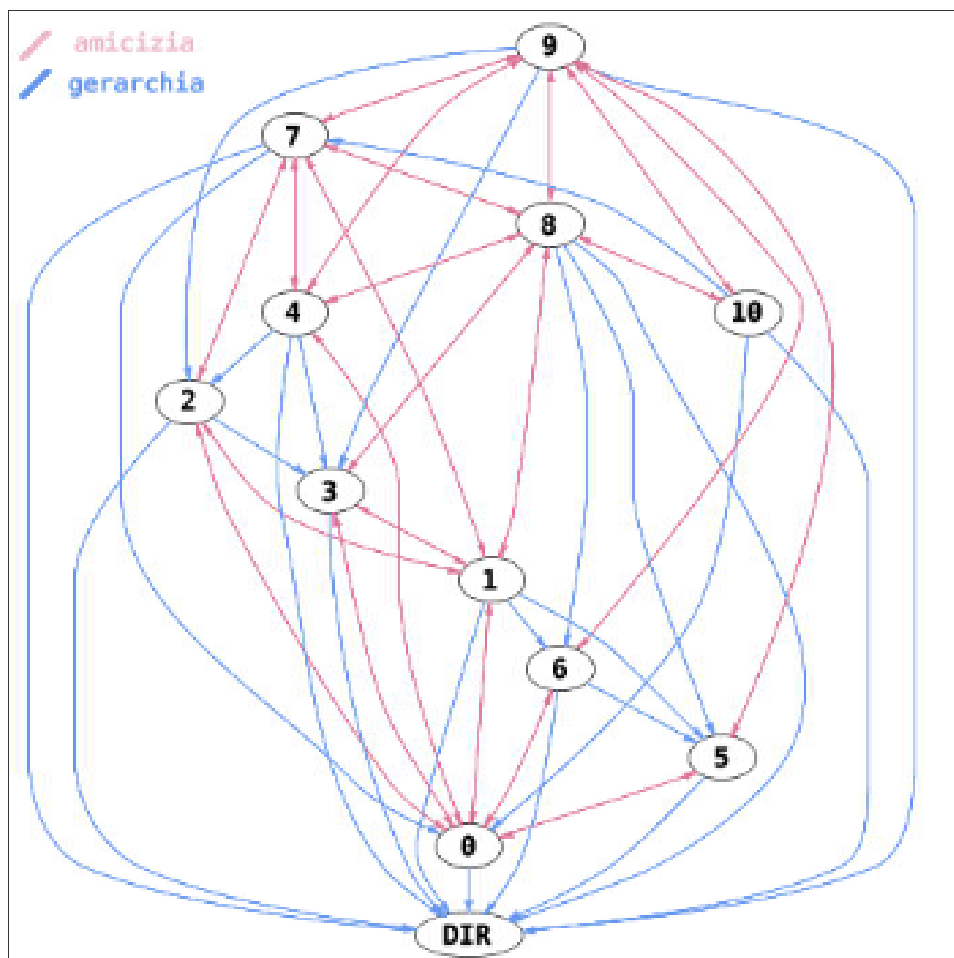
3. **ImpiegatiNoAmiciSettore:** un impiegato non può essere amico di un manager che lavora nello stesso settore
4. **SuperioriStessoSettore:** se un dipendente A è superiore di un dipendente B allora questi lavorano nello stesso settore

- Simulazione: grazie alla simulazione abbiamo verificato che nello scope considerato è possibile trovare un'istanza plausibile del modello specificato

Tutti i test sopra descritti sono stati effettuati con uno scope pari a 12 elementi

Risultati del model checking

Per prima cosa sono state valutate le quattro asserzioni per dimostrare la validità di alcuni principi cui il modello deve sottostare per essere corretto. Per ogni asserzione, Alloy non è stato in grado di trovare controesempi nello scope specificato, per cui si è passati ad analizzare una simulazione del sistema per vedere se Alloy era in grado di trovare almeno un'istanza valida del modello così specificato. L'istanza che il programma ha identificato è presente nell'Appendice del presente documento.



Nel diagramma soprastante sono riportate le relazioni **gerarchia** (freccie blu) ed **amicizia** (freccia rosa) tra i dipendenti dell'azienda (identificati dai numeri) ed il dirigente (identificato da DIR). Come è lecito aspettarsi, si nota che tutte le frecce rappresentano relazioni di amicizia sono bidirezionali (rispecchiando quindi la simmetria insita nella relazione stessa). Inoltre non è possibile identificare rapporti di amicizia tra dipendenti già legati da una relazione di gerarchia (non ci sono frecce blu tra elementi collegati da una freccia rosa). Il caso del dirigente è il caso limite: egli non ha amici in azienda, essendo il superiore di tutti gli altri dipendenti (e forse non è nemmeno umano).

Trattiamo ora una visuale tabellare dei dati raccolti per l'istanza rilevata dal programma:

	D0M	D1F	D2I	D3M	D4F	D5M	D6I	D7I	D8F	D9F	D10F
D0M		A	A	A	A	A	A				
D1F	A		A	A		S	S	A	A		
D2I	A	A		S				A			
D3M	A	A							A		
D4F	A		S	S				A	A	A	
D5M	A									A	
D6I	A					S				A	
D7I	S	A	A		A				A	A	
D8F		A		A	A	S	S	A		A	
D9F		S	S		A	A	A	A	A		A
D10F	S							S	A	A	

Legenda

- **D[numero][lettera]** : il numero identifica il dipendente, la lettera è l'iniziale della mansione
- **Verde: settore 0 – Rosso : settore 1 – Blu : settore 2**
- **A** : relazione di amicizia tra i dipendenti
- **S** : il dipendente nella riga è sottoposto al dipendente nella colonna

Analizzando in dettaglio la tabella soprastante si scopre che nell'istanza sono verificate tutte le proprietà di rilievo nel nostro modello (è stata omessa dalla tabella la figura del dirigente in quanto poco significativa). In particolare:

- Ogni settore dell'azienda (Rosso,Blu,Verde) ha almeno tre dipendenti, uno per tipologia
- Le relazioni di amicizia sono riflessive (Simmetria sulla relazione **A**)

- E'possibile l'amicizia tra dipendenti quadri inferiori e superiori purchè in settori diversi (ad esempio il fattorino D4 del settore Blu è amico dell'impiegato D7 del settore Rosso)
- I manager non hanno mai amici all'interno del proprio settore, nè hanno superiori (escludendo i dirigenti dell'azienda)

Sono stati ripetuti diversi cicli di simulazione, in particolare si è provato a imporre un numero di dirigenti pari al numero di settori dell'azienda e le cose sono andate nel modo previsto e confacente ai risultati ottenuti nella precedente trattazione. Ad ogni modo non è stato possibile variare troppo le condizioni per la scarsa numerosità imposta dai limiti computazionali della macchina di simulazione.

Conclusioni

In definitiva, l'esperimento ha dimostrato che è possibile garantire la correttezza della specifica di un sistema informativo come quello richiesto, perlomeno nell'ambito degli scope simulati durante l'esperienza.

Utilizzando risorse di calcolo più potenti, si ritiene che sarà possibile validare il modello anche per numerosità più vicine a quelle tipiche di un sistema aziendale di questo tipo. Alternativamente sarebbe possibile cercare ulteriori ottimizzazioni del modello in modo da ridurre la complessità computazionale.

Appendice

A) Istanza di riferimento identificata da Alloy durante lo svolgimento dell'esperimento

```
module alloy/lang/univ
```

```
sig univ = {AZIENDA_0, DIPENDENTE_0, DIPENDENTE_1, DIPENDENTE_10, DIPENDENTE_2, DIPENDENTE_3,
DIPENDENTE_4, DIPENDENTE_5, DIPENDENTE_6, DIPENDENTE_7, DIPENDENTE_8, DIPENDENTE_9, DIRIGENTE_0,
FATTORINO_0, IMPIEGATO_0, MANAGER_0, RITIRO_0, SETTORE_0, SETTORE_1, SETTORE_2, SPEDIZIONE_0,
VERIFICA_0}
```

```
module azienda
```

```
sig AZIENDA extends univ = AZIENDA_0
```

```
  dirigenti : some azienda/DIRIGENTE =
    {AZIENDA_0 -> DIRIGENTE_0}
```

```
  settori : some azienda/SETTORE =
```

```

{AZIENDA_0 -> {SETTORE_0, SETTORE_1, SETTORE_2}}
sig INDIVIDUO extends univ = {DIPENDENTE_0, DIPENDENTE_1, DIPENDENTE_10, DIPENDENTE_2, DIPENDENTE_3,
DIPENDENTE_4, DIPENDENTE_5, DIPENDENTE_6, DIPENDENTE_7, DIPENDENTE_8, DIPENDENTE_9, DIRIGENTE_0}
  amici : set azienda/INDIVIDUO =
    {DIPENDENTE_0 -> {DIPENDENTE_1, DIPENDENTE_2, DIPENDENTE_3, DIPENDENTE_4, DIPENDENTE_5,
DIPENDENTE_6},
      DIPENDENTE_1 -> {DIPENDENTE_0, DIPENDENTE_2, DIPENDENTE_3, DIPENDENTE_7, DIPENDENTE_8},
      DIPENDENTE_10 -> {DIPENDENTE_8, DIPENDENTE_9},
      DIPENDENTE_2 -> {DIPENDENTE_0, DIPENDENTE_1, DIPENDENTE_7},
      DIPENDENTE_3 -> {DIPENDENTE_0, DIPENDENTE_1, DIPENDENTE_8},
      DIPENDENTE_4 -> {DIPENDENTE_0, DIPENDENTE_7, DIPENDENTE_8, DIPENDENTE_9},
      DIPENDENTE_5 -> {DIPENDENTE_0, DIPENDENTE_9},
      DIPENDENTE_6 -> {DIPENDENTE_0, DIPENDENTE_9},
      DIPENDENTE_7 -> {DIPENDENTE_1, DIPENDENTE_2, DIPENDENTE_4, DIPENDENTE_8, DIPENDENTE_9},
      DIPENDENTE_8 -> {DIPENDENTE_1, DIPENDENTE_10, DIPENDENTE_3, DIPENDENTE_4, DIPENDENTE_7,
DIPENDENTE_9},
      DIPENDENTE_9 -> {DIPENDENTE_10, DIPENDENTE_4, DIPENDENTE_5, DIPENDENTE_6, DIPENDENTE_7,
DIPENDENTE_8}}
sig DIPENDENTE extends INDIVIDUO = {DIPENDENTE_0, DIPENDENTE_1, DIPENDENTE_10, DIPENDENTE_2,
DIPENDENTE_3, DIPENDENTE_4, DIPENDENTE_5, DIPENDENTE_6, DIPENDENTE_7, DIPENDENTE_8, DIPENDENTE_9}
  mansione : one azienda/MANAGER + azienda/IMPIEGATO + azienda/FATTORINO =
    {DIPENDENTE_0 -> MANAGER_0,
      DIPENDENTE_1 -> FATTORINO_0,
      DIPENDENTE_10 -> FATTORINO_0,
      DIPENDENTE_2 -> IMPIEGATO_0,
      DIPENDENTE_3 -> MANAGER_0,
      DIPENDENTE_4 -> FATTORINO_0,
      DIPENDENTE_5 -> MANAGER_0,
      DIPENDENTE_6 -> IMPIEGATO_0,
      DIPENDENTE_7 -> IMPIEGATO_0,
      DIPENDENTE_8 -> FATTORINO_0,
      DIPENDENTE_9 -> FATTORINO_0}
  settore_impiego : one azienda/SETTORE =
    {DIPENDENTE_0 -> SETTORE_0,
      DIPENDENTE_1 -> SETTORE_1,
      DIPENDENTE_10 -> SETTORE_0,
      DIPENDENTE_2 -> SETTORE_2,
      DIPENDENTE_3 -> SETTORE_2,
      DIPENDENTE_4 -> SETTORE_2,
      DIPENDENTE_5 -> SETTORE_1,
      DIPENDENTE_6 -> SETTORE_1,
      DIPENDENTE_7 -> SETTORE_0,
      DIPENDENTE_8 -> SETTORE_1,
      DIPENDENTE_9 -> SETTORE_2}
  superiori : some azienda/DIRIGENTE + azienda/DIPENDENTE =
    {DIPENDENTE_0 -> DIRIGENTE_0,
      DIPENDENTE_1 -> {DIPENDENTE_5, DIPENDENTE_6, DIRIGENTE_0},
      DIPENDENTE_10 -> {DIPENDENTE_0, DIPENDENTE_7, DIRIGENTE_0},

```

```

DIPENDENTE_2 -> {DIPENDENTE_3, DIRIGENTE_0},
DIPENDENTE_3 -> DIRIGENTE_0,
DIPENDENTE_4 -> {DIPENDENTE_2, DIPENDENTE_3, DIRIGENTE_0},
DIPENDENTE_5 -> DIRIGENTE_0,
DIPENDENTE_6 -> {DIPENDENTE_5, DIRIGENTE_0},
DIPENDENTE_7 -> {DIPENDENTE_0, DIRIGENTE_0},
DIPENDENTE_8 -> {DIPENDENTE_5, DIPENDENTE_6, DIRIGENTE_0},
DIPENDENTE_9 -> {DIPENDENTE_2, DIPENDENTE_3, DIRIGENTE_0}}
sig DIRIGENTE extends INDIVIDUO = {DIRIGENTE_0}
sig FATTORINO extends univ = FATTORINO_0
sig IMPIEGATO extends univ = IMPIEGATO_0
sig MANAGER extends univ = MANAGER_0
sig SETTORE extends univ = {SETTORE_0, SETTORE_1, SETTORE_2}
  tipologia : one azienda/RITIRO + azienda/VERIFICA + azienda/SPEDIZIONE =
    {SETTORE_0 -> VERIFICA_0,
     SETTORE_1 -> RITIRO_0,
     SETTORE_2 -> SPEDIZIONE_0}
sig RITIRO extends univ = RITIRO_0
sig VERIFICA extends univ = VERIFICA_0
sig SPEDIZIONE extends univ = SPEDIZIONE_0

```

B) Istruzioni Alloy commentate del modello utilizzato durante l'esperimento

module azienda

```

// =====
//                SIGNATURES
// =====

```

```

// definizione tipo azienda
// un'azienda è guidata da almeno un dirigente
// ed è composta da alcuni settori (nel nostro caso pari a tre)
one sig AZIENDA{
    dirigenti      :      some DIRIGENTE,
    settori        :      some SETTORE
}

```

```

// definizione tipo individuo
// ogni individuo ha un insieme di amici
abstract sig INDIVIDUO{
    amici          :      set INDIVIDUO
}

```

```

// definizione dipendente a partire dall'individuo
// il dipendente ha una mansione, un settore associato di impiego
// e ha associati alcuni superiori

```

```

sig DIPENDENTE extends INDIVIDUO{
    mansione           :           one (MANAGER + IMPIEGATO + FATTORINO),
    settore_impiego    :           one SETTORE,
    superiori          :           some (DIRIGENTE + DIPENDENTE)
}

// definizione dirigente
// la specifica fornita non vincola il dirigente ad un particolare settore
// per cui non è stato implementato alcun attributo in grado di
// correlare un dirigente ad un particolare settore nonostante essi
// siano coordinatori dei settori stessi. Si suppone che l'insieme di dirigenti
// rappresenti un "collegio direttivo" dell'azienda che prende decisioni
// in un unico team.
abstract sig DIRIGENTE extends INDIVIDUO{}

// definizione impieghi possibili all'interno
// di un settore
one abstract sig FATTORINO{}
one abstract sig IMPIEGATO{}
one abstract sig MANAGER{}

// definizione dei settori lavorativi
// Ogni settore ha una propria tipologia associata
sig SETTORE{
    tipologia          :           one (RITIRO + VERIFICA + SPEDIZIONE)
}

// definizione delle tipologie dei settori dell'azienda.
one abstract sig RITIRO{}
one abstract sig VERIFICA{}
one abstract sig SPEDIZIONE{}

// =====
//                FATTI
// =====

// L'azienda considerata ha esattamente tre settori, uno per
// ciascuna tipologia definita (ritiro, verifica, spedizione)
// Supponiamo che la simulazione consideri una sola azienda
// con i tre settori sopra descritti. La cardinalità dell'azienda è già
// stata definita nella signature (tramite il prefisso one), occorre quindi
// definire esplicitamente la cardinalità dei settori nel fatto

fact SettoriPresenti {SETTORE = AZIENDA.settori && #SETTORE=3}

```

```

fact TipologieSettori {one settore1: SETTORE | one settore2: SETTORE | one settore3: SETTORE |
settore1.tipologia = RITIRO &&
settore2.tipologia = VERIFICA &&
settore3.tipologia = SPEDIZIONE &&
settore1 != settore2 && settore1 != settore3 && settore2 != settore3
}

// Ogni dipendente presente deve essere associato ad un solo settore in quanto non
// ha senso che lavori contemporaneamente in più settori. In questo modo si assicura
// anche che ogni dipendente abbia effettivamente una sola mansione associata
// poichè la dichiarazione delle mansioni prevede una scelta disgiuntiva tra le tre
// definite. Inoltre questo vincolo impone che tutti i dipendenti presenti non siano disoccupati

fact UnSoloSettore {
    all dip : DIPENDENTE | one sett : SETTORE | dip.settore_impiego = sett
}

// Supponiamo che in ogni settore sia presente almeno un dipendente per ogni
// mansione considerata

fact AlmenoUnDipendentePerMansione {
    all sett : SETTORE | some dip1 : DIPENDENTE, dip2: DIPENDENTE, dip3: DIPENDENTE |
    dip1.mansione = FATTORINO && dip2.mansione = IMPIEGATO && dip3.mansione = MANAGER
    && dip1 != dip2 && dip1 != dip3 && dip2 != dip3
    && dip1.settore_impiego = sett && dip2.settore_impiego = sett && dip3.settore_impiego = sett
}

// Vincoli di gerarchia tra i dipendenti
// =====
// Un dipendente non può essere superiore di se stesso (no proprietà riflessiva)
// Se il dipendente A è superiore del dipendente B non è possibile che B sia superiore di A (no proprietà simmetrica)
// Se il dipendente A è superiore del dipendente B, il quale a sua volta è superiore del dipendente C
// allora A è superiore di C (proprietà transitiva)
// Queste proprietà vengono intrinsecamente specificate nei fatti sotto definiti, poichè si fornisce una regola
// di comportamento per ogni caso specifico, definendo in modo assoluto gli elementi che appartengono
// all'insieme dei superiori per ogni dipendente presente nella simulazione
// I fattorini avranno come superiori impiegati e manager appartenenti al loro settore lavorativo, più tutti i dirigenti

fact GerarchiaFattorini {
    all sett: SETTORE , dip1 : DIPENDENTE, dip2 : DIPENDENTE, dip3 : DIPENDENTE |
    (dip1.settore_impiego = sett && dip2.settore_impiego = sett && dip3.settore_impiego = sett &&
    dip1.mansione = FATTORINO && dip2.mansione = IMPIEGATO && dip3.mansione = MANAGER)
    =>
    ((dip2+dip3+DIRIGENTE) = dip1.superiori)
}

```

```
// Gli impiegati avranno come superiori i manager appartenenti al loro settore lavorativo, più tutti i dirigenti
```

```
fact GerarchiaImpiegati{
    all sett: SETTORE, dip1 : DIPENDENTE, dip2 : DIPENDENTE |
    (dip1.settore_impiego = sett && dip2.settore_impiego = sett &&
    dip1.mansione = IMPIEGATO && dip2.mansione = MANAGER)
    =>
    (dip2+DIRIGENTE = dip1.superiori)
}
```

```
// I manager avranno come superiori tutti i dirigenti dell'azienda.
```

```
fact GerarchiaManager{
    all dip1: DIPENDENTE |
    (dip1.mansione = MANAGER)
    =>
    (DIRIGENTE = dip1.superiori)
}
```

```
// Vincoli di amicizia tra due individui
```

```
// =====
```

```
// Un individuo non può essere amico di se stesso (no proprietà riflessiva)
```

```
// Se A è amico di B, allora B è amico di A (proprietà simmetrica)
```

```
// Se A è amico di B e B è amico di C non è detto che A sia amico di C (no proprietà transitiva)
```

```
fact AmiciziaNonRiflessiva {
    all ind: INDIVIDUO | ind !in ind.amici
}
```

```
fact AmiciziaSimmetrica{
    all indA: INDIVIDUO, indB : INDIVIDUO | indA in indB.amici => indB in indA.amici
}
```

```
// Dati un dipendente ed un individuo, se il dipendente è tra gli amici dell'individuo
```

```
// allora l'individuo non è tra i superiori del dipendente
```

```
fact NoAmicoTraISuperiori {
    all dip : DIPENDENTE | no ind : INDIVIDUO |
    dip in ind.amici && ind in dip.superiori
}
```

```

// =====
//                               ASSERT
// =====

// Condizioni di controllo che devono essere verificate nel modello sopra specificato

// I dirigenti non possono avere amici tra i dipendenti

assert DirettoriNoAmici {
    no dip: DIPENDENTE | some dir : DIRIGENTE | dir in dip.amici
}

// I fattorini non possono avere amici tra i manager e gli impiegati del proprio settore

assert FattoriniNoAmiciSettore{
    no dip1: DIPENDENTE | some dip2: DIPENDENTE |
    dip1.settore_impiego = dip2.settore_impiego &&
    dip1.mansione = FATTORINO &&
    dip2.mansione != FATTORINO &&
    dip1 in dip2.amici
}

// Un impiegato non può essere amico di un manager che lavora nello stesso settore

assert ImpiegatiNoAmiciSettore{
    no dip1: DIPENDENTE | some dip2: DIPENDENTE |
    dip1.settore_impiego = dip2.settore_impiego &&
    dip1.mansione = IMPIEGATO &&
    dip2.mansione = MANAGER &&
    dip1 in dip2.amici
}

// Se un dipendente A è superiore di un dipendente B allora questi lavorano nello stesso settore

assert SuperioriStessoSettore{
    no dip1: DIPENDENTE | some dip2: DIPENDENTE |
    dip1 in dip2.superiori &&
    dip1.settore_impiego != dip2.settore_impiego
}

```

```
// =====  
//                                MODEL CHECKING  
// =====  
  
// Controllo delle Assert sopra definite  
  
check DirettoriNoAmici for 12  
check FattoriniNoAmiciSettore for 12  
check ImpiegatiNoAmiciSettore for 12  
check SuperioriStessoSettore for 12  
  
// Ricerca di un'istanza plausibile del sistema  
  
pred Simulazione () {  
    some DIRIGENTE && some DIPENDENTE && some SETTORE  
}  
  
// Esecuzione della ricerca  
  
run Simulazione for 12
```