UNIVERSITAT POLITÈCNICA DE CATALUNYA

DEPARTAMENT DE LLENGUATGES I SISTEMES INFORMÀTICS

CARME QUER i BOSOR

# DETERMINACIÓ D'ESQUEMES D'INTERACCIÓ EN MODELS CONCEPTUALS ORIENTATS A OBJECTES AMB INTERACCIÓ IMPLÍCITA

TESI DOCTORAL

DIRIGIDA PEL DR. ANTONI OLIVÉ i RAMON

BARCELONA

1998

# Annex

Aquesta secció mostra la síntesi d'elements feta a partir del model Especificació del sistema exemple (secció 4.2).

**Classes d'esdeveniments externs**
EsdExterns = {crearBorsa, crearSector, crearAcció, crearVigilant, canviPreu, elimina}

**Classes d'objectes**
Objectes = {mercatBorsari, observable, acció, accióInicial, accióValorada, sector, borsa, canviAcció, vigilant, vigilantActiu, vigilantPreparat, vigilantActivat}

**Atributs d'una classe d'objectes**
Atributs(mercatBorsari) = {mercatBorsariBorsa}
Atributs(observable) = {observableMoviment, observableVigilant}
Atributs(acció) = {accióPreu, accióCanviAcció, accióSector, accióMoviment, accióVigilant}
Atributs(accióInicial) = {accióInicialPreu, accióInicialCanviAcció, accióInicialSector, accióInicialMoviment, accióInicialVigilant}
Atributs(accióValorada) = {accióValoradaPreu, accióValoradaCanviAcció, accióValoradaSector, accióValoradaMoviment, accióValoradaVigilant}
Atributs(sector) = {sectorBorsa, sectorAcció, sectorMoviment, sectorVigilant}
Atributs(borsa) = {borsaSector, mercatBorsari, borsaMoviment, borsaVigilant}
Atributs(canviAcció) = {canviAccióPreu, data, hora, canviAccióAcció}
Atributs(vigilant) = {vigilantLímit, vigilantDescripció, vigilantVigilat}
Atributs(vigilantActiu) = {vigilantActiuLímit, vigilantActiuDescripció, vigilantActiuVigilat}
Atributs(vigilantPreparat) = {vigilantPreparatLímit, vigilantPreparatDescripció, vigilantPreparatVigilat}
Atributs(vigilantActivat) = {vigilantActivatLímit, vigilantActivatDescripció, vigilantActivatVigilat}

**Classes d'esdeveniments generats associades a una classe d'objectes**
Generacions(mercatBorsari) = {newBorsa, newSector, newAcció, newVigilant}
Generacions(observable) = { }
Generacions(acció) = {accióComprovaPreu, newCanviAcció, newAccioObservable}
Generacions(accióInicial) = { }
Generacions(acciónValorada) = { }
Generacions(sector) = {sectorComprovaPreu, newSectorObservable}
Generacions(borsa) = {borsaComprovaPreu, newBorsaObservable}
Generacions(canviAcció) = { }
Generacions(vigilant) = {alarmaVigilant}
Generacions(vigilantActiu) = { }
Generacions(vigilantPreparat) = { }
Generacions(vigilantActivat) = { }

**Restriccions associades a una classe d'objectes**
Restriccions(mercatBorsari) = { }
Restriccions(observable) = { }
Restriccions(acció) = { }
Restriccions(accióInicial) = { }
Restriccions(acciónValorada) = { }
Restriccions(sector) = { }
Restriccions(borsa) = { }
Restriccions(canviAcció) = { }
Restriccions(vigilant) = {restricció$_1$} /* límit > 0 */
Restriccions(vigilantActiu) = { }
Restriccions(vigilantPreparat) = { }
Restriccions(vigilantActivat) = { }

Cal recordar que les restriccions donades pel fet que cap atribut en el model del sistema exemple té definit l'invariant "optional" no les considerarem.

**Classes d'esdeveniments estructurals d'inserció d'objectes**
Inserció(observable) = inserir_observable
Inserció(acció) = inserir_acció
Inserció(accióInicial) = inserir_accióInicial
Inserció(acciónValorada) = inserir_acciónValorada
Inserció(sector) = inserir_sector
Inserció(borsa) = inserir_borsa
Inserció(canviAcció) = inserir_canviAcció
Inserció(vigilant) = inserir_vigilant
Inserció(vigilantActiu) = inserir_vigilantActiu
Inserció(vigilantPreparat) = inserir_vigilantPreparat
Inserció(vigilantActivat) = inserir_vigilantActivat

**Classes d'esdeveniments estructurals d'esborrat d'objectes**
Esborrat(observable) = esborrar_observable
Esborrat(acció) = esborrar_acció
Esborrat(accióInicial) = esborrar_accióInicial
Esborrat(acciónValorada) = esborrar_acciónValorada
Esborrat(sector) = esborrar_sector
Esborrat(borsa) = esborrar_borsa
Esborrat(canviAcció) = esborrar_canviAcció
Esborrat(vigilant) = esborrar_vigilant
Esborrat(vigilantActiu) = esborrar_vigilantActiu
Esborrat(vigilantPreparat) = esborrar_vigilantPreparat
Esborrat(vigilantActivat) = esborrar_vigilantActivat

**Classes d'esdeveniments estructurals de modificació dels valors d'atributs**

Modificació(mercatBorsariBorsa) = {afegir_mercatBorsariBorsa, eliminar_mercatBorsariBorsa}
Modificació(observableMoviment) = {assignar_observableMoviment}
Modificació(observableVigilant) = {afegir_observableVigilant, eliminar_observableVigilant}
Modificació(accióPreu) = {assignar_accióPreu}
Modificació(accióCanviAcció) = {afegir_accióCanviAcció, eliminar_accióCanviAcció}
Modificació(accióSector) = {assignar_accióSector}
Modificació(accióMoviment) = {assignar_accióMoviment}
Modificació(accióVigilant) = {afegir_accióVigilant, eliminar_accióVigilant}
Modificació(accióInicialPreu) = {assignar_accióInicialPreu}
Modificació(accióInicialCanviAcció) = {afegir_accióInicialCanviAcció, eliminar_accióInicialCanviAcció}
Modificació(accióInicialSector) = {assignar_accióInicialSector}
Modificació(accióInicialMoviment) = {assignar_accióInicialMoviment}
Modificació(accióInicialVigilant) = {afegir_accióInicialVigilant, eliminar_accióInicialVigilant}
Modificació(accióValoradaPreu) = {assignar_accióValoradaPreu}
Modificació(accióValoradaCanviAcció) = {afegir_accióValoradaCanviAcció, eliminar_accióValoradaCanviAcció}
Modificació(accióValoradaSector) = {assignar_accióValoradaSector}
Modificació(accióValoradaMoviment) = {assignar_accióValoradaMoviment}
Modificació(accióValoradaVigilant) = {afegir_accióValoradaVigilant, eliminar_accióValoradaVigilant}
Modificació(sectorBorsa) = {assignar_sectorBorsa}
Modificació(sectorAcció) = {afegir_sectorAcció,eliminar_sectorAcció}
Modificació(sectorMoviment) = {assignar_sectorMoviment}
Modificació(sectorVigilant) = {afegir_sectorVigilant, eliminar_sectorVigilant}
Modificació(borsaSector) = {afegir_borsaSector, eliminar_borsaSector}
Modificació(mercatBorsari) = {assignar_mercatBorsari}
Modificació(borsaMoviment) = {assignar_borsaMoviment}
Modificació(borsaVigilant) = {afegir_borsaVigilant, eliminar_borsaVigilant}
Modificació(canviAccióPreu) = {assignar_canviAccióPreu}
Modificació(data) = {assignar_data}
Modificació(hora) ={assignar_hora}
Modificació(canviAccióAcció) = {assignar_canviAccióAcció}
Modificació(vigilantLímit) = {assignar_vigilantLímit}
Modificació(vigilantDescripció) = {assignar_vigilantDescripció}
Modificació(vigilantVigilat) = {assignar_vigilantVigilat}
Modificació(vigilantActiuLímit) = {assignar_vigilantActiuLímit}
Modificació(vigilantActiuDescripció) = {assignar_vigilantActiuDescripció}
Modificació(vigilantActiuVigilat) = {assignar_vigilantActiuVigilat}
Modificació(vigilantPreparatLímit) = {assignar_vigilantPreparatLímit}
Modificació(vigilantPreparatDescripció) = {assignar_vigilantPreparatDescripció}
Modificació(vigilantPreparatVigilat) = {assignar_vigilantPreparatVigilat}
Modificació(vigilantActivatLímit) = {assignar_vigilantActivatLímit}
Modificació(vigilantActivatDescripció) = {assignar_vigilantActivatDescripció}
Modificació(vigilantActivatVigilat) = {assignar_vigilantActivatVigilat}

**Classes d'esdeveniments estructurals de generació d'esdeveniments**

Generació(newBorsa) = newBorsa
Generació(newSector) = newSector

Generació(newAcció) = newAcció
Generació(newVigilant) = newVigilant
Generació(accióComprovaPreu) = accióComprovaPreu
Generació(sectorComprovaPreu) = sectorComprovaPreu
Generació(borsaComprovaPreu) = borsaComprovaPreu
Generació(newCanviAcció) = newCanviAcció
Generació(newAccioObservable) = newAccioObservable
Generació(newSectorObservable) = newSectorObservable
Generació(newBorsaObservable) = newBorsaObservable
Generació(alarmaVigilant) = alarmaVigilant

**Classes d'esdeveniments estructurals de violació de restriccions d'integritat**

Restricció(restricció$_1$) = restricció$_1$ /* límit > 0 */

**Regles d'esdeveniments que es refereixen a classes d'esdeveniments estructurals d'inserció d'objectes**

Regles(inserir_observable) = {$r_1$,$r_2$,$r_3$}
causa($r_1$) = {newAccioObservable}
prerequisit($r_1$) = { }
causa($r_2$) = {newSectorObservable}
prerequisit($r_2$) = { }
causa($r_3$) = {newBorsaObservable}
prerequisit($r_3$) = { }
Regles(inserir_acció) = {$r_4$}
causa($r_4$) = {newAcció}
prerequisit($r_4$) = { }
Regles(inserir_accióInicial) = {$r_5$}
causa($r_5$) = {newAcció}
prerequisit($r_5$) = { }
Regles(inserir_accióValorada) = {$r_6$}
causa($r_6$) = {canviPreu}
prerequisit($r_6$) = { }
Regles(inserir_sector) = {$r_7$}
causa($r_7$) = {newSector}
prerequisit($r_7$) = { }
Regles(inserir_borsa) = {$r_8$}
causa($r_8$) = {newBorsa}
prerequisit($r_8$) = { }
Regles(inserir_canviAcció) = {$r_9$}
causa($r_9$) = {newCanviAcció}
prerequisit($r_9$) = { }
Regles(inserir_vigilant) = {$r_{10}$}
causa($r_{10}$) = {newVigilant}
prerequisit($r_{10}$) = { }
Regles(inserir_vigilantActiu) = {$r_{11}$}
causa($r_{11}$) = {newVigilant}
prerequisit($r_{11}$) = { }
Regles(inserir_vigilantPreparat) = {$r_{12}$, $r_{13}$,$r_{14}$,$r_{15}$}
causa($r_{12}$) = {newVigilant}
prerequisit($r_{12}$) = { }
causa($r_{13}$) = {accióComprovaPreu}
prerequisit($r_{13}$) = {assignar_accióPreu,afegir_accióCanviAcció,
assignar_accióMoviment, inserir_accióValorada,
esborrar_accióInicial,assignar_observableMoviment,

assignar_acciónicialMoviment,assignar_sectorMoviment,
assignar_acciónaloradaMoviment,
afegir_acciónicialCanviAcció, afegir_acciónaloradaCanviAcció,
assignar_canviAcciónAcció,assignar_acciónicialPreu,
assignar_acciónaloradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}

causa($r_{14}$) = {sectorComprovaPreu}

prerequisit($r_{14}$) = {assignar_acciónPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_acciónicial,assignar_observableMoviment,
assignar_acciónicialMoviment,assignar_sectorMoviment,
assignar_acciónaloradaMoviment,
afegir_acciónicialCanviAcció, afegir_acciónaloradaCanviAcció,
assignar_canviAcciónAcció,assignar_acciónicialPreu,
assignar_acciónaloradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}

causa($r_{15}$) = {borsaComprovaPreu}

prerequisit($r_{15}$) = {assignar_acciónPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_acciónicial,assignar_observableMoviment,
assignar_acciónicialMoviment,assignar_sectorMoviment,
assignar_acciónaloradaMoviment,
afegir_acciónicialCanviAcció, afegir_acciónaloradaCanviAcció,
assignar_canviAcciónAcció,assignar_acciónicialPreu,
assignar_acciónaloradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}

Regles(inserir_vigilantActivat) = {$r_{16}$,$r_{17}$,$r_{18}$}

causa($r_{16}$) = {accióComprovaPreu}

prerequisit($r_{16}$) = {assignar_acciónPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_acciónicial,assignar_observableMoviment,
assignar_acciónicialMoviment,assignar_sectorMoviment,
assignar_acciónaloradaMoviment,
afegir_acciónicialCanviAcció, afegir_acciónaloradaCanviAcció,
assignar_canviAcciónAcció,assignar_acciónicialPreu,
assignar_acciónaloradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}

causa($r_{17}$) = {sectorComprovaPreu}

prerequisit($r_{17}$) = {assignar_acciónPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_acciónicial,assignar_observableMoviment,
assignar_acciónicialMoviment,assignar_sectorMoviment,
assignar_acciónaloradaMoviment,
afegir_acciónicialCanviAcció, afegir_acciónaloradaCanviAcció,
assignar_canviAcciónAcció,assignar_acciónicialPreu,
assignar_acciónaloradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}

causa($r_{18}$) = {borsaComprovaPreu}

prerequisit($r_{18}$) = {assignar_acciónPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_acciónicial,assignar_observableMoviment,
assignar_acciónicialMoviment,assignar_sectorMoviment,
assignar_acciónaloradaMoviment,

afegir_accióInicialCanviAcció, afegir_acciónValoradaCanviAcció,
assignar_canviAcciónAcció,assignar_accióInicialPreu,
assignar_acciónValoradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}

**Regles d'esdeveniments que es refereixen a classes d'esdeveniments estructurals d'esborrat d'objectes**

Regles(esborrar_observable) = { }
Regles(esborrar_acció) = { }
Regles(esborrar_accióInicial) = {$r_{19}$}
causa($r_{19}$) = {canviPreu}
prerequisit($r_{19}$) = { }
Regles(esborrar_acciónValorada) = { }
Regles(esborrar_sector) = { }
Regles(esborrar_borsa) = { }
Regles(esborrar_canviAcció) = { }
Regles(esborrar_vigilant) = {$r_{20}$}
causa($r_{20}$) = {elimina}
prerequisit($r_{20}$) = { }
Regles(esborrar_vigilantActiu) = {$r_{21}$}
causa($r_{21}$) = {elimina}
prerequisit($r_{21}$) = { }
Regles(esborrar_vigilantPreparat) = {$r_{22}$, $r_{23}$,$r_{24}$,$r_{25}$}
causa($r_{22}$) = {elimina}
prerequisit($r_{22}$) = { }
causa($r_{23}$) = {accióComprovaPreu}
prerequisit($r_{23}$) = {assignar_accióPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_accióInicial,assignar_observableMoviment,
assignar_accióInicialMoviment,assignar_sectorMoviment,
assignar_acciónValoradaMoviment,
afegir_accióInicialCanviAcció, afegir_acciónValoradaCanviAcció,
assignar_canviAcciónAcció,assignar_accióInicialPreu,
assignar_acciónValoradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}
causa($r_{24}$) = {sectorComprovaPreu}
prerequisit($r_{24}$) = {assignar_accióPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_accióInicial,assignar_observableMoviment,
assignar_accióInicialMoviment,assignar_sectorMoviment,
assignar_acciónValoradaMoviment,
afegir_accióInicialCanviAcció, afegir_acciónValoradaCanviAcció,
assignar_canviAcciónAcció,assignar_accióInicialPreu,
assignar_acciónValoradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}
causa($r_{25}$) = {borsaComprovaPreu}
prerequisit($r_{25}$) = {assignar_accióPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_accióInicial,assignar_observableMoviment,
assignar_accióInicialMoviment,assignar_sectorMoviment,
assignar_acciónValoradaMoviment,
afegir_accióInicialCanviAcció, afegir_acciónValoradaCanviAcció,
assignar_canviAcciónAcció,assignar_accióInicialPreu,

assignar_accióValoradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}

Regles(esborrar_vigilantActivat) = {$r_{26}$,$r_{27}$,$r_{28}$,$r_{29}$}
causa($r_{26}$) = {elimina}
prerequisit($r_{26}$) = { }
causa($r_{27}$) = {acciónComprovaPreu}
prerequisit($r_{27}$) = {assignar_acciónPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_acciónInicial,assignar_observableMoviment,
assignar_acciónInicialMoviment,assignar_sectorMoviment,
assignar_acciónValoradaMoviment,
afegir_acciónInicialCanviAcció, afegir_acciónValoradaCanviAcció,
assignar_canviAcciónAcció,assignar_acciónInicialPreu,
assignar_acciónValoradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}
causa($r_{28}$) = {sectorComprovaPreu}
prerequisit($r_{28}$) = {assignar_acciónPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_acciónInicial,assignar_observableMoviment,
assignar_acciónInicialMoviment,assignar_sectorMoviment,
assignar_acciónValoradaMoviment,
afegir_acciónInicialCanviAcció, afegir_acciónValoradaCanviAcció,
assignar_canviAcciónAcció,assignar_acciónInicialPreu,
assignar_acciónValoradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}
causa($r_{29}$) = {borsaComprovaPreu}
prerequisit($r_{29}$) = {assignar_acciónPreu,afegir_acciónCanviAcció,
assignar_acciónMoviment, inserir_acciónValorada,
esborrar_acciónInicial,assignar_observableMoviment,
assignar_acciónInicialMoviment,assignar_sectorMoviment,
assignar_acciónValoradaMoviment,
afegir_acciónInicialCanviAcció, afegir_acciónValoradaCanviAcció,
assignar_canviAcciónAcció,assignar_acciónInicialPreu,
assignar_acciónValoradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora}

**Regles d'esdeveniments que es refereixen a classes d'esdeveniments estructurals de modificació dels valors d'atributs**

Regles(afegir_mercatBorsariBorsa) = {$r_{30}$}
causa($r_{30}$) = {crearBorsa}
prerequisit($r_{30}$) = { }
Regles(eliminar_mercatBorsariBorsa) = { }
Regles(assignar_observableMoviment) = {$r_{31}$,$r_{32}$,$r_{33}$}
causa($r_{31}$) = {assignar_acciónMoviment}
prerequisit($r_{31}$) = {newCanviAcció, inserir_canviAcció,
assignar_canviAcciónPreu, assignar_data, assignar_hora,
newAccioObservable, inserir_observable}
causa($r_{32}$) = {assignar_sectorMoviment}
prerequisit($r_{32}$) = {newSectorObservable, inserir_observable}
causa($r_{33}$) = {assignar_borsaMoviment}
prerequisit($r_{33}$) = {newBorsaObservable, inserir_observable}
Regles(afegir_observableVigilant) = {$r_{34}$}

causa($r_{34}$) = {assignar_vigilantVigilat}
prerequisit($r_{34}$) = { }
Regles(eliminar_observableVigilant) = {$r_{35}$}
causa($r_{35}$) = {assignar_vigilantVigilat}
prerequisit($r_{35}$) = { }
Regles(assignar_acciónPreu) = {$r_{36}$}
causa($r_{36}$) = {canviPreu}
prerequisit($r_{36}$) = { }
Regles(afegir_accióCanviAcció) = {$r_{37}$}
causa($r_{37}$) = {canviPreu}
prerequisit($r_{37}$) = { }
Regles(eliminar_accióCanviAcció) = { }
Regles(assignar_acciónSector) = {$r_{38}$}
causa($r_{38}$) = {newAcció}
prerequisit($r_{38}$) = { }
Regles(assignar_acciónMoviment) = {$r_{39}$,$r_{40}$}
causa($r_{39}$) = {canviPreu}
prerequisit($r_{39}$) = { }
causa($r_{40}$) = {newAcció}
prerequisit($r_{40}$) = { }
Regles(afegir_acciónVigilant) = {$r_{41}$}
causa($r_{41}$) = {assignar_vigilantVigilat}
prerequisit($r_{41}$) = { }
Regles(eliminar_acciónVigilant) = {$r_{42}$}
causa($r_{42}$) = {assignar_vigilantVigilat}
prerequisit($r_{42}$) = { }
Regles(assignar_accióInicialPreu) = {$r_{43}$}
causa($r_{43}$) = {assignar_acciónPreu}
prerequisit($r_{43}$) = {newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora}
Regles(afegir_accióInicialCanviAcció) = {$r_{44}$}
causa($r_{44}$) = {afegir_accióCanviAcció}
prerequisit($r_{44}$) = {newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora}
Regles(eliminar_accióInicialCanviAcció) = {$r_{45}$}
causa($r_{45}$) = {eliminar_accióCanviAcció}
prerequisit($r_{45}$) = { }
Regles(assignar_accióInicialSector) = {$r_{46}$}
causa($r_{46}$) = {assignar_acciónSector}
prerequisit($r_{46}$) = {newAccioObservable, inserir_observable}
Regles(assignar_accióInicialMoviment) = {$r_{47}$}
causa($r_{47}$) = {assignar_acciónMoviment}
prerequisit($r_{47}$) = {newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora,
newAccioObservable, inserir_observable}
Regles(afegir_accióInicialVigilant) = {$r_{48}$}
causa($r_{48}$) = {afegir_acciónVigilant}
prerequisit($r_{48}$) = { }
Regles(eliminar_accióInicialVigilant) = {$r_{49}$}
causa($r_{49}$) = {eliminar_acciónVigilant}
prerequisit($r_{49}$) = { }
Regles(assignar_acciónValoradaPreu) = {$r_{50}$}
causa($r_{50}$) = {assignar_acciónPreu}
prerequisit($r_{50}$) = {newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora}
Regles(afegir_acciónValoradaCanviAcció) = {$r_{51}$}
causa($r_{51}$) = {afegir_accióCanviAcció}
prerequisit($r_{51}$) = {newCanviAcció, inserir_canviAcció,

assignar_canviAcciónPreu, assignar_data, assignar_hora}
Regles(eliminar_accióValoradaCanviAcció) = {$r_{52}$}
causa($r_{52}$) = {eliminar_accióCanviAcció}
prerequisit($r_{52}$) = { }
Regles(assignar_accióValoradaSector) = {$r_{53}$}
causa($r_{53}$) = {assignar_accióSector}
prerequisit($r_{53}$) = {newAccioObservable, inserir_observable}
Regles(assignar_accióValoradaMoviment) = {$r_{54}$}
causa($r_{54}$) = {assignar_accióMoviment}
prerequisit($r_{54}$) = {newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora,
newAccioObservable, inserir_observable}
Regles(afegir_accióValoradaVigilant) = {$r_{55}$}
causa($r_{55}$) = {afegir_acciónVigilant}
prerequisit($r_{55}$) = { }
Regles(eliminar_accióValoradaVigilant) = {$r_{56}$}
causa($r_{56}$) = {eliminar_acciónVigilant}
prerequisit($r_{56}$) = { }
Regles(assignar_sectorBorsa) = {$r_{57}$}
causa($r_{57}$) = {newSector}
prerequisit($r_{57}$) = { }
Regles(afegir_sectorAcció) = {$r_{58}$}
causa($r_{58}$) = {assignar_accióSector}
prerequisit($r_{58}$) = {newAccioObservable, inserir_observable}
Regles(eliminar_sectorAcció) = {$r_{59}$}
causa($r_{59}$) = {assignar_accióSector}
prerequisit($r_{59}$) = {newAccioObservable, inserir_observable}
Regles(assignar_sectorMoviment) = {$r_{60}$,$r_{61}$}
causa($r_{60}$) = {assignar_acciónMoviment}
prerequisit($r_{60}$) = {newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora,
newAccioObservable, inserir_observable}
causa($r_{61}$) = {newSector}
prerequisit($r_{61}$) = { }
Regles(afegir_sectorVigilant) = {$r_{62}$}
causa($r_{62}$) = {assignar_vigilantVigilat}
prerequisit($r_{62}$) = { }
Regles(eliminar_sectorVigilant) = {$r_{63}$}
causa($r_{63}$) = {assignar_vigilantVigilat}
prerequisit($r_{63}$) = { }
Regles(afegir_borsaSector) = {$r_{64}$}
causa($r_{64}$) = {assignar_sectorBorsa}
prerequisit($r_{64}$) = {newSectorObservable,inserirObservable}
Regles(eliminar_borsaSector) = {$r_{65}$}
causa($r_{65}$) = {assignar_sectorBorsa}
prerequisit($r_{65}$) = {newSectorObservable,inserirObservable}
Regles(assignar_mercatBorsari) = {$r_{66}$}
causa($r_{66}$) = {newBorsa}
prerequisit($r_{66}$) = { }
Regles(assignar_borsaMoviment) = {$r_{67}$,$r_{68}$}
causa($r_{67}$) = {assignar_sectorMoviment}
prerequisit($r_{67}$) = {newSectorObservable, inserir_observable}
causa($r_{68}$) = {newBorsa}
prerequisit($r_{68}$) = { }
Regles(afegir_borsaVigilant) = {$r_{69}$}
causa($r_{69}$) = {assignar_vigilantVigilat}
prerequisit($r_{69}$) = { }
Regles(eliminar_borsaVigilant) = {$r_{70}$}

causa($r_{70}$) = {assignar_vigilantVigilat}
prerequisit($r_{70}$) = { }
Regles(assignar_canviAccióPreu) = {$r_{71}$}
causa($r_{71}$) = {newCanviAcció}
prerequisit($r_{71}$) = { }
Regles(assignar_data) = {$r_{72}$}
causa($r_{72}$) = {newCanviAcció}
prerequisit($r_{72}$) = { }
Regles(assignar_hora) = {$r_{73}$}
causa($r_{73}$) = {newCanviAcció}
prerequisit($r_{73}$) = { }
Regles(assignar_canviAccióAcció) = {$r_{74}$,$r_{75}$}
causa($r_{74}$) = {afegir_accióCanviAcció}
prerequisit($r_{74}$) = {newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora}
causa($r_{75}$) = {eliminar_accióCanviAcció}
prerequisit($r_{75}$) = { }
Regles(assignar_vigilantLímit) = {$r_{76}$}
causa($r_{76}$) = {newVigilant}
prerequisit($r_{76}$) = { }
Regles(assignar_vigilantDescripció) = {$r_{77}$}
causa($r_{77}$) = {newVigilant}
prerequisit($r_{77}$) = { }
Regles(assignar_vigilantVigilat) = {$r_{78}$}
causa($r_{78}$) = {newVigilant}
prerequisit($r_{78}$) = { }
Regles(assignar_vigilantActiuLímit) = {$r_{79}$}
causa($r_{79}$) = {assignar_vigilantLímit}
prerequisit($r_{79}$) = { }
Regles(assignar_vigilantActiuDescripció) = {$r_{80}$}
causa($r_{80}$) = {assignar_vigilantDescripció}
prerequisit($r_{80}$) = { }
Regles(assignar_vigilantActiuVigilat) = {$r_{81}$}
causa($r_{81}$) = {assignar_vigilantVigilat}
prerequisit($r_{81}$) = { }
Regles(assignar_vigilantPreparatLímit) = {$r_{82}$}
causa($r_{82}$) = {assignar_vigilantActiuLímit}
prerequisit($r_{82}$) = { }
Regles(assignar_vigilantPreparatDescripció) = {$r_{83}$}
causa($r_{83}$) = {assignar_vigilantActiuDescripció}
prerequisit($r_{83}$) = { }
Regles(assignar_vigilantPreparatVigilat) = {$r_{84}$}
causa($r_{84}$) = {assignar_vigilantActiuVigilat}
prerequisit($r_{84}$) = { }
Regles(assignar_vigilantActivatLímit) = {$r_{85}$}
causa($r_{85}$) = {assignar_vigilantActiuLímit}
prerequisit($r_{85}$) = { }
Regles(assignar_vigilantActivatDescripció) = {$r_{86}$}
causa($r_{86}$) = {assignar_vigilantActiuDescripció}
prerequisit($r_{86}$) = { }
Regles(assignar_vigilantActivatVigilat) = {$r_{87}$}
causa($r_{87}$) = {assignar_vigilantActiuVigilat}
prerequisit($r_{87}$) = { }

**Regles d'esdeveniments que es refereixen a classes d'esdeveniments estructurals de violació de restriccions d'integritat**

Regles(restricció$_1$) = {$r_{88}$}
causa($r_{88}$) = {assignar_vigilantLímit}
prerequisit($r_{88}$) = { }

**Regles d'esdeveniments que es refereixen a classes d'esdeveniments estructurals de generació d'esdeveniments**

Regles(newAccioObservable) = {$r_{89}$}
causa($r_{89}$) = {newAcció}
prerequisit($r_{89}$) = { }
Regles(newSectorObservable) = {$r_{90}$}
causa($r_{90}$) = {newSector}
prerequisit($r_{90}$) = { }
Regles(newBorsaObservable) = {$r_{91}$}
causa($r_{91}$) = {newBorsa}
prerequisit($r_{91}$) = { }
Regles(newBorsa) = {$r_{92}$}
causa($r_{92}$) = {crearBorsa}
prerequisit($r_{92}$) = { }
Regles(newSector) = {$r_{93}$}
causa($r_{93}$) = {crearSector}
prerequisit($r_{93}$) = { }
Regles(newAcció) = {$r_{94}$}
causa($r_{94}$) = {crearAcció}
prerequisit($r_{94}$) = { }
Regles(newVigilant) = {$r_{95}$}
causa($r_{95}$) = {crearVigilant}
prerequisit($r_{95}$) = { }
Regles(accióComprovaPreu) = {$r_{96}$}
causa($r_{96}$) = {canviPreu}
prerequisit($r_{96}$) = { }
Regles(sectorComprovaPreu) = {$r_{97}$}
causa($r_{97}$) = {canviPreu}
prerequisit($r_{97}$) = { }
Regles(borsaComprovaPreu) = {$r_{98}$}
causa($r_{98}$) = {canviPreu}
prerequisit($r_{98}$) = { }
Regles(newCanviAcció) = {$r_{99}$}
causa($r_{99}$) = {canviPreu}
prerequisit($r_{99}$) = { }
Regles(alarmaVigilant) = {$r_{100}$,$r_{101}$,$_{102}$}
causa($r_{100}$) = {accióComprovaPreu}
prerequisit($r_{100}$) = {assignar_accióPreu,afegir_accióCanviAcció,
assignar_accióMoviment, inserir_accióValorada,
esborrar_accióInicial,assignar_observableMoviment,
assignar_accióInicialMoviment,assignar_sectorMoviment,
assignar_accióValoradaMoviment,
afegir_accióInicialCanviAcció, afegir_accióValoradaCanviAcció,
assignar_canviAccióAcció,assignar_accióInicialPreu,
assignar_accióValoradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora}
causa($r_{101}$) = {sectorComprovaPreu}
prerequisit($r_{101}$) = {assignar_accióPreu,afegir_accióCanviAcció,
assignar_accióMoviment, inserir_accióValorada,

esborrar_accióInicial,assignar_observableMoviment,
assignar_accióInicialMoviment,assignar_sectorMoviment,
assignar_acciónValoradaMoviment,
afegir_accióInicialCanviAcció, afegir_acciónValoradaCanviAcció,
assignar_canviAccióAcció,assignar_accióInicialPreu,
assignar_acciónValoradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora}

causa($r_{102}$) = {borsaComprovaPreu}

prerequisit($r_{102}$) = {assignar_accióPreu,afegir_accióCanviAcció,
assignar_accióMoviment, inserir_acciónValorada,
esborrar_accióInicial,assignar_observableMoviment,
assignar_accióInicialMoviment,assignar_sectorMoviment,
assignar_acciónValoradaMoviment,
afegir_accióInicialCanviAcció, afegir_acciónValoradaCanviAcció,
assignar_canviAccióAcció,assignar_accióInicialPreu,
assignar_acciónValoradaPreu, assignar_borsaMoviment,
newCanviAcció, inserir_canviAcció,
assignar_canviAccióPreu, assignar_data, assignar_hora}

# 5. Aplicació als MCDO

En aquest capítol descriurem com aplicar el mètode per determinar Esquemes d'Interacció a models conceptuals deductius orientats a objectes (MCDO). En parlar dels MCDO no pressuposem un llenguatge concret, sinó models amb unes certes característiques que poden estar escrits en diversos llenguatges. Per qüestions pràctiques, utilitzarem un llenguatge genèric que permet expressar tots els components d'aquests models. També mostrarem alguns Esquemes obtinguts mitjançant el nostre mètode a partir de la síntesi d'elements en un MCDO.

El capítol es divideix en cinc seccions. A la secció 5.1 introduirem els components d'un MCDO i el llenguatge genèric utilitzat. A la secció 5.2 veurem un exemple de l'ús d'aquest llenguatge per modelar un sistema concret. A la secció 5.3 mostrarem com fer la síntesi dels elements necessaris per a la determinació d'Esquemes d'Interacció. I per últim, a la secció 5.4, donarem alguns exemples d'Esquemes d'Interacció, obtinguts mitjançant el nostre mètode, per al cas del model exemple de la secció 5.2.

## 5.1 Introducció als MCDO

En l'enfocament deductiu [Oli86,San94] es modela la part estàtica dels sistemes d'informació mitjançant predicats i la part dinàmica mitjançant regles. El problema principal d'aquest enfocament és la manca d'un mecanisme d'estructuració de la part estàtica. Els MCDO, per resoldre aquest problema, adopten el mecanisme orientat a objectes d'estructuració de la informació.

Concretament, agrupen en classes d'objectes els predicats d'un model que fan referència a objectes d'una mateixa classe, i en classes d'esdeveniments els predicats que fan referència a esdeveniments d'una mateixa classe. Els aspectes estàtic i dinàmic del sistema queden barrejats dins d'aquestes unitats d'estructuració dels models.

Com en els models amb un enfocament deductiu pur, en els MCDO el temps és un concepte clau. Són models en què són accessibles tots els estats del sistema des de l'inici. Aquests estats s'identifiquen per un instant de temps, que s'expressa en una unitat bàsica de temps (segon, minut, hora) prou fina perquè no es produeixin ambigüitats en les informacions. Així, $T = \{t_i,...,t_f\}$ és el conjunt ordenat de temps consecutius que identifiquen els diferents estats pels quals ha passat el sistema. En aquest conjunt cada instant està expressat en la unitat bàsica de temps i $t_i$ i $t_f$ identifiquen l'instant corresponent a l'estat inicial i l'estat final del sistema.

A les dues subseccions que trobem tot seguit veurem com definir classes d'esdeveniments externs i classes d'objectes en un MCDO.

### 5.1.1 Classes d'esdeveniments externs

Els esdeveniments externs corresponen a canvis en el domini que són notificats al sistema d'informació. Les classes d'esdeveniments externs agrupen tota la informació sobre els esdeveniments que notifiquen un mateix tipus de canvi.

En els MCDO es permet que en un instant determinat es pugui comunicar al sistema més d'un esdeveniment extern de la mateixa classe o de classes diferents. D'altra banda, en els MCDO que nosaltres considerarem, tots els canvis d'estat que es produeixin han de ser causats per l'ocurrència d'esdeveniments externs. És a dir, no considerarem MCDO, amb comportament "espontani", en què puguin haver-hi canvis d'estat causats pel simple pas del temps.

La definició d'una classe d'esdeveniments externs en el llenguatge genèric que usarem en aquesta tesi, és la que segueix:

**Event class** esdev(eid:Esdev;[$atr_1$:$d_1$;..;$atr_j$:$d_j$;] time:Time
[**derived** $atr_{j+1}$:$d_{j+1}$;..;$atr_n$:$d_n$])
<regles atributs derivats>
[ **constraints** <definició restriccions d'integritat> ]
**End**

**Atributs**

En aquesta definició, *esdev* és una classe d'esdeveniments externs amb atributs $eid, atr_1 ... atr_n$, i amb dominis en què aquests atributs prenen els seus valors $Esdev, d_1, ..., d_n$.

Totes les classes d'esdeveniments externs tenen els atributs *eid* i *time*, que són atributs als quals assigna valor el mateix sistema en el moment en què ocorre un esdeveniment extern:

- El valor de l'atribut *eid* permet identificar els esdeveniments externs d'una mateixa classe. *Esdev* és, per tant, el domini dels identificadors interns dels esdeveniments de la classe *esdev*.

- El valor de l'atribut *time* correspon a l'instant de temps en què ocorre l'esdeveniment. El domini *Time* és la unitat de temps bàsica que s'ha establert per al sistema que es modela.

Aquests atributs, com que els tenen totes les classes d'esdeveniments externs, poden no explicitar-se en la seva definició.

La resta d'atributs són particulars de les diferents classes d'esdeveniments externs, i poden ser de dos tipus, segons com prenguin valor:

- $atr_1 ... atr_j$ són atributs que prenen valor de l'entorn.

- i $atr_{j+1} ... atr_n$ són atributs derivats el valor dels quals es calcula quan ocorre un esdeveniment de la classe.

Els dominis en què prenen valors són $d_1 ... d_n$ i poden ser dominis predefinits simples (Integer, Real, String, Boolean, enumeracions d'enters, reals i de strings, Time) i dominis definits en el mateix MCDO simples (identificadors interns d'objectes d'una certa classe d'objectes).

**Exemple 5.1.1**

Una classe d'esdeveniments externs en un sistema de gestió de vendes d'una empresa és *nouPreu*. Els atributs necessaris per als esdeveniments d'aquesta classe són l'article a què s'ha canviat el preu i el nou preu:

**Event class** nouPreu (art:Article; preu:Integer) **End**

Tant l'atribut *art* com l'atribut *preu* són atributs que prenen valor de l'entorn. El domini de l'atribut *preu* és el domini predefinit simple *Integer*; en canvi, el domini de l'atribut *art* estarà definit en el mateix MCDO mitjançant una classe d'objectes *article*. Cal notar que els atributs *eid* i *time* no apareixen explícitament en aquesta definició. ♦

En el cas que els esdeveniments d'una classe tinguin atributs derivats, cal indicar, quin és el valor que prenen aquests atributs quan ocorre un esdeveniment de la classe. Això es fa definint el valor de cadascun amb una regla de deducció.

<regles atributs derivats> :== $atr_{j+1}(Z{:}VA_{j+1}) \leftarrow esdev(Z,T), W_{j+1}.$
:
:
$atr_n(Z{:}VA_n) \leftarrow esdev(Z,T), W_n.$

Les regles de deducció d'atributs derivats tenen sempre la forma: $atr_i(Z{:}VA_i) \leftarrow esdev(Z,T), W_i$. L'àtom al cap de la regla té dues variables: la primera, Z, fa referència a l'identificador intern de l'esdeveniment, i la segona, $VA_i$, al valor que pren l'atribut $atr_i$. El cos de les regles és una conjunció del literal *esdev(Z,T)*, que fa referència a l'ocurrència d'un esdeveniment amb *eid* Z de la classe per a la qual es defineix l'atribut i una fórmula de primer ordre $W_i$ (més endavant veurem quins són els predicats que poden aparèixer en aquestes fórmules). Si es vol, no cal explicitar el literal *esdev(Z,T)* al cos de la regla.

**Exemple 5.1.2**

El mateix sistema de gestió de vendes de l'exemple anterior pot necessitar una classe d'esdeveniments externs *venda*. Els atributs d'un esdeveniment d'aquesta classe són: l'article venut, el magatzem que l'ha venut, la quantitat d'unitats venudes, el client a qui s'han venut i l'import de la venda:

**Event class** venda (art: Article; mgt: Magatzem;
cli: Client; qtt: Integer; **derived** import: Integer).
import(Z:I) ← art(Z:A), qtt(Z:Q), article(A,T),
preuArt(A,T:Pr), I = Q*Pr.
**End**

L'atribut *import* és l'únic que és derivat. La seva regla de deducció defineix quin valor pren aquest atribut quan un esdeveniment *venda* amb *eid* Z ocorre en un instant T. En aquest cas, el literal *venda(Z,T)* no apareix explícitament en el cos de la regla. ♦

Abans de continuar, cal fer notar que:

- Per simplificar l'escriptura de les regles de deducció i restriccions d'integritat, el llenguatge genèric que estem introduint utilitza comes "," per indicar l'operador lògic "∧" i punts i comes ";" per indicar l'operador lògic "∨".

- Les regles de deducció dels MCDO que considerarem han de ser restringides temporalment al passat. És a dir que, cadascun dels predicats als quals es fa referència

al cos de les regles han d'estar definits sobre un instant anterior o igual a l'instant per al qual està definit l'àtom al cap de la regla.

- Les regles de deducció dels MCDO que considerarem no han de presentar recursivitat directa ni indirecta. Un conjunt de regles presenten recursivitat directa si el predicat en l'àtom del cap d'una de les regles apareix també al seu cos, i recursivitat indirecta si el predicat en l'àtom del cap d'una de les regles apareix al cos d'una segona regla del conjunt que defineix de manera directa o indirecta un dels predicats al cos de la primera.

**Restriccions d'integritat**

En el cas que hi hagi restriccions d'integritat molt relacionades amb l'ocurrència d'esdeveniments externs d'una classe, aquestes restriccions es defineixen en aquesta classe. Quan es produeix un d'aquests esdeveniments, s'han de satisfer les seves restriccions d'integritat, altrament l'esdeveniment serà rebutjat pel sistema.

<definició restriccions d'integritat> :==

$$\begin{aligned} &\text{incons\_esdev}_1(\mathbf{X_1}) \leftarrow \text{esdev}(Z,T),W_1. \\ &\vdots \\ &\text{incons\_esdev}_n(\mathbf{X_n}) \leftarrow \text{esdev}(Z,T),W_n. \end{aligned}$$

Tal com descriu [LIT84], tota restricció d'integritat es pot expressar sempre en forma de denegació. En el llenguatge genèric per a MCDO que aquí utilitzem, les restriccions d'integritat s'expressen en aquesta forma, associant-hi un predicat d'inconsistència per donar aparença de regla de deducció.

La definició d'una restricció d'integritat té sempre la forma: $\text{incons\_esdev}_i(\mathbf{X_i}) \leftarrow \text{esdev}(Z,T),W_i$. El predicat en l'àtom al cap de la regla és el predicat d'inconsistència *incons_esdev*$_i$, on $\mathbf{X_i}$ és un vector de variables que donaran informació sobre la causa per la qual es viola la restricció d'integritat. El cos de la regla és una conjunció entre el literal *esdev(Z,T)* que fa referència a l'ocurrència d'un esdeveniment amb *eid* Z de la classe per a la qual es defineix la restricció i una fórmula de primer ordre $W_i$ (més endavant veurem quins són els predicats que poden aparèixer en aquestes fórmules). Si es vol, no cal explicitar el literal *esdev(Z,T)* en el cos de la restricció.

**Exemple 5.1.3**

A la classe d'esdeveniments *venda* de l'exemple anterior, podríem associar-hi la restricció d'integritat *estocInsuficient*, que es violaria en el cas de notificar una venda d'una quantitat d'article superior a l'estoc d'aquest article al magatzem on es fa la venda.

**Event class** venda (art: ARTICLE; mgt: MAGATZEM;
cli: CLIENT; qtt: INTEGER; **derived** import: INTEGER).
:
**constraints**
estocInsuficient(A,M,Q,Qm,T) ← venda(Z:T),art(Z:A),mgt(Z:M),
qtt(Z:Q),estocMgt(M,T-1:[A,Qm]),Qm<Q.
**End** ♦

En la definició d'una classe d'esdeveniments externs cal una restricció d'integritat per a cadascun dels atributs, d'aquesta classe, que prenen un valor en un domini definit en el mateix MCDO. En aquestes restriccions s'exigeix que l'objecte, amb identificador intern el valor de l'atribut d'un esdeveniment de la classe, existeixi al sistema en l'instant de temps en què ocorre l'esdeveniment. Aquestes restriccions no cal definir-les explícitament.

**Exemple 5.1.4**

A part la restricció d'integritat *estocInsuficient*, la classe d'esdeveniments *venda* de l'exemple anterior té tres restriccions més, que són implícites en la seva definició. Els predicats d'inconsistència per a aquestes restriccions poden ser *clientInexistent*, *articleInexistent* i *magatzemInexistent*, i les definicions són, respectivament:

clientInexistent(C,T) ← venda(Z,T),cli(Z:C),not(client(C,T)).
articleInexistent(A,T) ← venda(Z,T),art(Z:A),not(article(A,T)).
magatzemInexistent(M,T) ← venda(Z,T),mgt(Z:M), not(magatzem(M,T)). ♦

Només hi ha un cas de classe d'esdeveniments externs en què aquesta restricció ha de ser menys estricta. Es tracta de les classes d'esdeveniments externs de destrucció d'objectes d'una classe d'objectes (vegeu més endavant). En aquestes classes hi ha sempre un atribut que pren un valor en un domini definit en el mateix MCDO. Concretament, l'atribut el valor del qual identifica l'objecte que s'eliminarà del sistema. La restricció per aquest atribut, ha d'exigir que l'objecte, amb identificador intern el valor de l'atribut, existeixi en el sistema en l'instant de temps anterior al de l'ocurrència de l'esdeveniment.

**Predicats en les regles de deducció i restriccions d'integritat**

Els predicats que apareixen al cos de les regles i restriccions d'integritat poden ser de tres tipus: predicats avaluables, predicats auxiliars predefinits i predicats definits en el mateix MCDO.

Els *predicats avaluables* són expressions aritmètiques o comparacions avaluables.

**Exemple 5.1.5**

A la regla de deducció del valor de l'atribut *import* d'una venda, exemple 5.1.2, hi ha el predicat avaluable I = Q*Pr, que és una expressió aritmètica. ♦

Els *predicats auxiliars predefinits* són quatre: max, min, count i sum.

- El predicat max(V,[C],R) retorna a R la instanciació amb més valor de la variable V que fa certa la condició C.

- El predicat min(V,[C],R) retorna a R la instanciació amb menys valor de la variable V que fa certa la condició C.

- El predicat count(V,[C],R) retorna a R el nombre de vegades que es pot fer certa la condició C, o sigui, el nombre de vegades que la variable V s'instància. Cal remarcar que el nombre resultant és el nombre d'instanciacions i no el nombre de valors diferents que es poden instanciar.

- Finalment, el predicat sum(V,[C],R) retorna a R la suma de les instanciacions de la variable V que fan certa la condició C.

En tots els casos C, és una fórmula de primer ordre en la qual apareix la variable V.

Els *predicats definits en el mateix MCDO* poden fer referència a diferents tipus d'informacions:

- Predicats que fan referència al valor d'un atribut d'un esdeveniment.

- Predicats que fan referència a l'existència d'un objecte en una classe en algun instant igual o anterior a l'instant en què ha ocorregut l'esdeveniment de la classe que es defineix.

- Predicats que fan referència al valor d'un atribut d'un objecte en un instant igual o anterior a l'instant en què ha ocorregut l'esdeveniment de la classe que es defineix.

- Predicats que fan referència a un esdeveniment ocorregut en un instant igual o anterior a l'instant en què ha ocorregut l'esdeveniment de la classe que es defineix.

**Exemple 5.1.6**

A la regla de deducció del valor de l'atribut derivat *import* de la classe d'esdeveniments externs *venda*, exemple 5.1.3, hi ha dos predicats que fan referència al valor de dos atributs de l'esdeveniment de la classe que es defineix:

```
import(Z:I) ← venda(Z,T), art(Z:A), qtt(Z:Q),
                    article(A,T), preuArt(A,T:Pr), I = Q*Pr.
```

Es tracta dels predicats *art* i *qtt*, que fan referència al valor dels atributs *art* i *qtt* de l'esdeveniment amb identificador intern Z de la classe d'esdeveniments *venda*. ♦

**Exemple 5.1.7**

A la mateixa regla de deducció, hi ha un predicat que fa referència a l'existència d'un objecte en una classe en el mateix instant en què ha ocorregut l'esdeveniment de la classe que es defineix. Es tracta del predicat *article*, que fa referència a l'existència de l'article A a la classe d'objectes *article* en l'instant en què ha ocorregut l'esdeveniment amb identificador intern Z de la classe *venda*. ♦

**Exemple 5.1.8**

També a la mateixa regla de deducció hi ha un predicat que fa referència al valor d'un atribut d'un objecte en el mateix instant en què ha ocorregut l'esdeveniment de la classe que es defineix. Es tracta del predicat *preuArt*, que fa referència al valor de l'atribut de nom *preuArt* de l'objecte amb identificador intern A de la classe *article* en el mateix instant T en què ha ocorregut l'esdeveniment amb identificador intern Z de la classe *venda*. ♦

**Exemple 5.1.9**

Finalment, per veure un exemple de predicat que fa referència a un esdeveniment ocorregut en un instant igual o anterior a l'instant en què ha ocorregut un esdeveniment de la classe que es defineix, afegirem una restricció d'integritat a la classe d'esdeveniments *venda*, exemple 5.1.3, per evitar que es venguin articles a aquells clients que en algun moment han estat declarats com a no solvents. La restricció és la següent:

$$\text{clientNoSolvent}(C,T) \leftarrow \text{venda}(Z,T),\text{cli}(Z{:}C),\ \text{noSolvent}(Y,T1),T1 \leq T,\text{cli}(Y{:}C).$$

El predicat *noSolvent* al cos d'aquesta nova restricció fa referència a l'ocurrència d'un esdeveniment amb identificador intern Y de la classe *noSolvent* en un instant anterior o igual a l'ocurrència de l'esdeveniment Z de la classe *venda*. ♦

### 5.1.2 Classes d'objectes

Les classes d'objectes estructuren la informació sobre els objectes al món real que són rellevants per al sistema d'informació. L'estat dels objectes canvia dinàmicament en el temps com a conseqüència dels canvis de l'entorn notificats pels esdeveniments externs. Cal recordar que en els MCDO són accessibles tots els estats del sistema des de l'inici, i per tant és accessible l'estat dels objectes en el sistema en qualsevol instant de temps.

**Classes extensionals, intensionals, generalitzacions i monoinstància**

Considerem que en un MCDO es poden definir quatre tipus de classes d'objectes: extensionals, intensionals, generalitzacions i monoinstància.

**Classes d'objectes extensionals.** Una classe extensional en un cert instant de temps T agrupa un conjunt d'objectes que han nascut en la classe en un instant anterior o igual a T i que encara no han mort. El naixement o la mort d'un objecte en una classe extensional és conseqüència directa de l'ocurrència d'un esdeveniment extern de creació o de destrucció. Quan un objecte neix en el sistema, ho fa sempre en una classe extensional, i en el mateix moment se li assigna un identificador intern que el diferencia de la resta d'objectes de la mateixa classe.

La sentència de definició d'una classe d'objectes extensional té la forma següent:

**Extensional object class** objecte
**created** objecte(oid:O,[$catr_1$:$A_1$,.,$catr_n$:$A_n$,] time:T) ←
cre_esdev(Z,T)[,$creatr_1$(Z:$A_1$),...,$creatr_n$(Z:$A_n$)].
[ **destroyed** objecte(oid:O,time:T) ← dse_esdev(Z,T), $dseatr_1$(Z:O).]
<cos classe d'objectes>
**End**

Tota classe extensional té una regla de creació. La regla de creació defineix la classe d'esdeveniments de naixement d'objectes a la classe. En la definició anterior el nom d'aquesta classe d'esdeveniments és *createdObjecte* i els seus atributs són *oid,*$catr_1$,...,$catr_n$,*time.* Cal notar que el domini d'aquests atributs no apareix en la definició de la classe, sinó que es dedueix del cos de la regla de creació en el cas dels atributs $catr_1$,...,$catr_n$, i és predefinit en el cas dels atributs *oid* i *time.* Així, O,$A_1$,...,$A_n$ no són dominis, sinó que són variables que prenen els valors dels diferents atributs.

Totes les classes d'esdeveniments de naixement d'objectes tenen els atributs *oid* i *time:*

- El valor de l'atribut *oid* permet identificar els objectes d'una mateixa classe. És un valor assignat pel sistema en el moment en què es crea un objecte, i s'usa també per identificar els esdeveniments de naixement d'objectes. Així, un mateix objecte no pot néixer dues vegades en la vida del sistema. El domini d'aquest atribut és el dels identificadors interns dels objectes de la classe *objecte.*

- El valor de l'atribut *time* correspon a l'instant de temps en què ha ocorregut l'esdeveniment extern de creació que ha provocat el naixement de l'objecte, i és un valor

que es dedueix de la definició de la regla de creació. El seu domini *Time* és la unitat de temps bàsica del sistema que es modela.

El cos d'una regla de creació té sempre la forma següent: "← cre_esdev(Z,T)[,creatr$_1$(Z:A$_1$),..., creatr$_n$(Z:A$_n$)]." Es tracta d'una conjunció del literal *cre_esdev(Z,T)* que fa referència a l'ocurrència de l'esdeveniment extern Z de creació d'objectes en la classe *objecte,* amb de 0 a n literals que fan referència al valor dels atributs d'aquest esdeveniment, *creatr$_1$,...,creatr$_n$*. La condició per a l'ocurrència d'un esdeveniment de naixement, *createdObjecte,* és sempre l'ocurrència d'un i només un esdeveniment extern de creació.

**Exemple 5.1.10**

Una classe d'objectes extensional, en el sistema de gestió de vendes que hem anat utilitzant en els exemples, és la classe *client.* La definició d'aquesta classe pot ser la següent:

**Extensional object class** client
**created** client(oid:C,tipusInicial:Ti,time:T) ← nouClient(Z,T), tipus(Z:Ti).
**End**

La classe d'esdeveniments de naixement d'objectes de la classe *client* és *createdClient,* i la classe d'esdeveniments externs de creació és *nouClient.* Observant la regla de creació, es pot veure que els valors dels atributs *tipusInicial* i *time* es deduiran a partir del cos de la regla. ♦

Una classe extensional pot no tenir definida una regla de destrucció. Això voldrà dir que els objectes de la classe mai no moren i, per tant, que a partir del naixement sempre existiran al sistema. La regla de destrucció defineix la classe d'esdeveniments de mort d'objectes de la classe. A la definició anterior, el nom d'aquesta classe d'esdeveniments és *destroyedObjecte,* i els seus atributs són *oid* i *time.* Tal com passa a la definició de les classes d'esdeveniments de naixement, O i T no són dominis dels atributs *oid* i *time,* sinó que són les variables que prenen els valors d'aquests atributs.

Totes les classes d'esdeveniments de mort d'objectes tenen aquests atributs i només aquests atributs:

- El valor de l'atribut *oid* és l'identificador intern de l'objecte que mor, i s'usa també per identificar els esdeveniments de mort d'objectes. El seu domini és el dels identificadors interns dels objectes de la classe *objecte.*

- El valor de l'atribut *time* correspon a l'instant de temps en què ha ocorregut l'esdeveniment extern de destrucció que ha provocat la mort de l'objecte. El seu domini *Time* és la unitat de temps bàsica del sistema que es modela.

El cos d'una regla de destrucció té sempre la forma següent: "← dse_esdev(Z,T), $dseatr_1$(Z:O).". Es tracta d'una conjunció del literal *dse_esdev(Z,T)* que fa referència a l'ocurrència d'un esdeveniment extern Z de destrucció de la classe *objecte* amb un literal que fa referència al valor de l'atribut d'aquest esdeveniment, el valor del qual identifica l'objecte que ha de morir, $dseatr_1$. La condició per a l'ocurrència d'un esdeveniment de mort, *destroyedObjecte*, és sempre l'ocurrència d'un i només un esdeveniment extern de destrucció.

**Exemple 5.1.11**

Una altra classe d'objectes necessària en el sistema de gestió de vendes és la classe *article*. La seva definició pot ser la següent:

**Extensional object class** article
  **created** article(oid:A,numArt:Na,preuInicial:Pr,time:T) ←
    nouArticle(Z,T), num(Z:Na),preu(Z:Pr).
  **destroyed** article(oid:A,time:T) ← esbArticle(Z,T),art(Z:A).
**End**

La classe d'esdeveniments de mort d'objectes de la classe *article* és *destroyedArticle* i la classe d'esdeveniments externs de destrucció és *esbArticle*. ♦

A la definició d'una classe extensional sempre hi ha implícita la regla de deducció associada al predicat que fa referència a l'existència d'un objecte en la classe en un cert instant. Aquesta regla té una forma o una altra segons la classe tingui definida o no una classe d'esdeveniments de mort:

objecte(O,T) ← **created** objecte(O,T1), T1≤T,
  not(**destroyed** object(O,T2),T2>T1,T2≤T).

objecte(O,T) ← **created** objecte(O,T1), T1≤T.

**Classes d'objectes intensionals.** Una classe intensional en un cert instant de temps agrupa aquells objectes que estan en totes les seves superclasses en el mateix instant i que tenen unes certes característiques. Les superclasses d'una classe intensional són classes intensionals i/o extensionals.

La sentència de definició d'una classe d'objectes intensional és:

**Intensional object class** subObjecte **is a** <llista de superclasses>
**derivation** subObjecte(O,T) ← $super_1$(O,T)[,...,$super_n$(O,T)],W.
<cos classe d'objectes>
**End**

Tota classe intensional té una regla de deducció que indica quins són els objectes que hi pertanyen en un cert instant de temps T. El cos d'una d'aquestes regles de deducció té sempre la forma següent: "← $super_1$(O,T) [,...,$super_n$(O,T)],W.". Es tracta d'una conjunció d'1 a N literals $super_i(O,T)$ que fan referència a la pertinença de l'objecte O a les seves superclasses en el mateix instant de temps i una fórmula de primer ordre W (més endavant veurem quins són els predicats que poden aparèixer en aquestes fórmules). Si els literals $super_i(O,T)$ no apareixen al cos de la regla, s'hi consideren implícits. Per saber quins són aquests literals, només cal mirar la llista de superclasses de la definició:

<llista de superclasses> :== $super_1$[,...,$super_n$]

**Exemple 5.1.12**
Una classe d'objectes intensional en el sistema de gestió de vendes és la classe *comestible*:

**Intensional object class** comestible **is a** article
**derivation** comestible(C,T) ← tipus(C,T: menjar).
**End**

Aquesta classe agrupa aquells articles que es poden menjar. El literal *article(C,T)* és implícit al cos de la regla de deducció. Un article serà de la classe *comestible,* si el valor de l'atribut *tipus* de l'article és *menjar.*♦

**Classes d'objectes generalització.** Una classe d'objectes generalització és una superclasse d'una o més classes extensionals i/o generalització. La població d'una classe generalització és la unió de tots els objectes que formen les seves subclasses.

La sentència de definició d'una classe d'objectes generalització té la forma següent:

**Generalization object class** superObjecte **of** <llista de subclasses>
**derivation** superObjecte(O,T) ← $sub_1$(O,T) [; ... ; $sub_n$(O,T)].
<cos classe d'objectes>
**End**

Tota classe generalització té una regla de deducció que defineix quins són els objectes que hi pertanyen en un cert instant de temps T. El cos d'una d'aquestes regles de deducció té sempre la forma següent: "← $sub_1$(O,T) [; ... ; $sub_n$(O,T)].". Es tracta d'una disjunció d'1 a N literals $sub_i(O,T)$ que fan referència a la pertinença de l'objecte a les seves subclasses. Si aquesta regla no apareix en la definició de la classe, s'hi considera implícita. Per saber quin ha de ser el cos de la regla només cal mirar la llista de subclasses de la definició:

<llista de subclasses> :== $sub_1$[,...,$sub_n$]

**Exemple 5.1.13**
Una classe d'objectes generalització al sistema de gestió de vendes és la classe *persona*:

**Generalization object class** persona **of** client, venedor
...
**End**

En aquest cas, la regla de pertinença implícita és:

**derivation** persona(C,T) ← client(C,T);venedor(C,T). ♦

**Classes d'objectes monoinstància.** Les classes monoinstància són un tipus de classe extensional amb un únic objecte. En aquest tipus de classe no hi ha naixement ni mort d'objectes, ja que l'objecte existeix a la classe en qualsevol instant passat, present i futur.

**Exemple 5.1.14**
Es pot ampliar el sistema de gestió de vendes amb la classe monoinstància *empresa*:

**Single object class** empresa
**End** ♦

**Atributs**
Una classe d'objectes pot tenir dos tipus d'atributs: atributs constants i atributs variables en el temps.

**[Extensional | Intensional | Generalized] object class** objecte
:
[ **constant attributes** <definició d'atributs constants>]
[ **time varying attributes** <definició d'atributs variables>]
[ **derivation rules** <regles de deducció d'atributs>]
:
**End**

**Atributs constants.** Els atributs constants prenen el seu valor en la creació de l'objecte, és a dir, en el naixement de l'objecte a la seva classe extensional, i conserven el valor mentre l'objecte no es destrueixi. Així, es pot dir que el valor de l'atribut no varia, independentment dels canvis que hi hagi durant la vida de l'objecte. La definició d'atributs constants té la forma següent:

<definició d'atributs constants> :== $atr\_const_1:d_1$;
:
$atr\_const_n:d_n$;

on $atr\_const_1,...,atr\_const_n$ són els noms dels atributs constants i $d_1,...,d_n$ els dominis en què prenen valor.

Els dominis en què aquests atributs prenen valor poden ser de dos tipus: dominis predefinits simples i dominis definits en el mateix MCDO simples.

Només es poden definir atributs constants a les classes extensionals i a les classes generalització. Els objectes a les classes intensionals hereten els definits a les seves superclasses. La definició de quin valor prenen els atributs constants dels objectes d'una classe es dóna a la regla de creació d'aquests objectes.

**Exemple 5.1.15**

Suposant que als articles al sistema de gestió de vendes, se'ls assigna un número d'article, cal definir un atribut constant *numArt* per als objectes de la classe d'objectes extensional *article*, exemple 5.1.11. La regla de creació de la classe defineix quin serà el valor de l'atribut:

**Extensional object class** article
**created** article(oid:A,numArt:Na,preuInicial:Pr,time:T) ←
nouArticle(Z,T), num(Z:Na),preu(Z:Pr).
:
**constant attributes** numArt: Integer.
**End** ♦

**Exemple 5.1.16**

D'altra banda, suposem un atribut constant *nom* dels objectes de la classe generalització *persona.* Aquest atribut ha de prendre el seu valor en el naixement dels objectes de la classe *persona* a la subclasse extensional *client.* Perquè sigui així, la definició de la regla de creació de la classe *client* ha de donar valor a l'atribut *nom* de la classe *persona.* La definició de les classes *persona* i *client* serà la següent:

**Generalization object class** persona **of** Client
  **constant attributes** nom: STRING.
**End**

**Extensional object class** client
  **created** client(oid:C, nom: N, time:T) ← nouClient(Z,T), nom(Z:N).
**End** ♦

Les regles de deducció d'atributs constants són implícites en la definició de les classes. Una d'aquestes regles té una forma o una altra segons si el domini en què l'atribut constant pren valor és un domini predefinit simple o un domini definit en el mateix MCDO:

$atr\_const_i$(O,T:VA) ← objecte(O,T),
  **created** objecte(O,T1),T1≤T,atr(O:VA).

$atr\_const_i$(O,T:VA) ← objecte(O,T),objecte_dom(VA,T),
  **created** objecte(O,T1),T1≤T,atr(O:VA).

Ambdues regles indiquen que l'atribut *atr_const*$_i$ de l'objecte O de la classe *objecte* té per valor VA en l'instant T si l'objecte O existeix a la classe *objecte* en T, i en un instant T1 anterior o igual a T ha ocorregut un esdeveniment de naixement de l'objecte O que ha donat el valor VA a l'atribut. La diferència al cos de la segona regla és que s'exigeix que el valor de l'atribut *atr_const*$_i$ en un instant T sigui l'identificador intern d'un objecte de la classe *objecte_dom* en el mateix instant.

**Atributs variables en el temps.** Els atributs variables en el temps poden prendre, canviar o perdre el seu valor durant la vida de l'objecte. La definició d'atributs variables té la forma següent:

<definició d'atributs variables> :== $atr\_var_1:d_1$;
  :
  $atr\_var_n:d_n$;

on $atr\_var_1,...,atr\_var_n$ són els noms dels atributs variables i $d_1,...,d_n$ els dominis en què prenen valor.

Els dominis en què aquests atributs prenen valor poden ser de tres tipus: dominis predefinits simples, dominis definits en el mateix MCDO simples i dominis estructurats. Els dominis estructurats són:

- **record of** $elem_1:d_1,...,elem_n:d_n$. L'atribut que es defineix pren n valors en n dominis. Cadascun d'aquests valors té un significat determinat i diferenciat en l'atribut per les etiquetes $elem_1,...,elem_n$.
- **set of** d. L'atribut que es defineix pot tenir un conjunt de valors del domini d.
- **set of record of** $elem_1:d_1,...,elem_n:d_n$. Combina els dos anteriors.

Es poden definir atributs variables en qualsevol tipus de classe. La definició de com prenen, canvien o perden valor es dóna en la seva regla de deducció.

**Exemple 5.1.17**

Els articles al sistema de gestió de vendes tenen clarament dos atributs variables en el temps: *preuArt*, que correspon al preu de l'article i *estocGlobal*, que correspon a l'estoc que hi ha de l'article en el conjunt de tots els magatzems de l'empresa.

**Extensional object class** article
:
**time varying attributes** preuArt: INTEGER.
estocGlobal: INTEGER.
**derivation rules**
:
**End** ♦

Les regles de deducció d'atributs variables en el temps s'han d'explicitar a la definició de la classe d'objectes per a la qual estan definits:

<regles de deducció d'atributs> :== $atr\_obj_1(O,T:VA_1) \leftarrow objecte(O,T),W_1.$
:
$atr\_obj_i(O,T:[VA_{i1},..,VA_{ip}]) \leftarrow objecte(O,T),W_i.$
:
$atr\_obj_j(O,T:VA_j) \leftarrow objecte(O,T),objecte\_dom(VA_j:T),W_j.$
:
$atr\_obj_n(O,T:VA_n) \leftarrow objecte(O,T),W_n.$

Les variables en l'àtom al cap de les regles de deducció d'aquest tipus d'atributs varien segons si es tracta d'un atribut definit sobre un domini **record of** o no. En el primer cas, hi ha una única variable corresponent al valor de l'atribut (aquest és el cas

dels atributs $atr\_obj_1$, $atr\_obj_j$ i $atr\_obj_n$ a la definició anterior). En el segon cas, hi ha una variable per a cada element en el **record of** (aquest és el cas de l'atribut $atr\_obj_i$ a la definició anterior).

El cos de les regles de deducció d'atributs amb domini predefinit simple, d'una classe *objecte*, és una conjunció entre el literal *objecte(O,T)* que fa referència a l'existència de l'objecte O en l'instant T, i una fórmula de primer ordre (més endavant veurem quins són els predicats que poden aparèixer en aquesta fórmula). En el cas que l'atribut prengui valors en un domini definit en el mateix MCDO, en la conjunció del cos de la regla hi haurà un literal més, *objecte_dom(VA,T)*. Si es vol, no cal explicitar el literal *objecte(O,T)*, i en el seu cas el literal *objecte_dom(VA,T)*, al cos de la regla.

**Herència d'atributs.** Una classe d'objectes hereta els atributs definits a les seves superclasses directes o indirectes. Tal com es presenten en aquesta tesi els MCDO, no es permet la redefinició d'atributs ni tampoc definir dos atributs amb un mateix nom en dues classes d'objectes. Així, la regla de deducció del valor d'un atribut heretat és sempre implícita en un MCDO i es pot definir com s'indica tot seguit:

subclasseNomAtribut(O,T:VA) ← superclasseNomAtribut(O,T:VA).

**Classes d'esdeveniments generats**

Els objectes d'una classe d'objectes poden detectar circumstàncies en el seu estat que poden afectar l'estat d'altres objectes al sistema o a l'entorn del sistema. Aquestes circumstàncies s'anuncien en forma d'esdeveniments generats. Al cos d'una classe d'objectes es poden definir les classes d'esdeveniments que els objectes de la classe d'objectes poden generar:

**[Extensional | Intensional | Generalized] object class** objecte
:
[ **generated events** <definició esdeveniments generats> ]
**End**

Les circumstàncies que poden fer que s'hagi de generar un esdeveniment d'una certa classe s'indiquen en un model mitjançant regles de deducció:

<definició esdeveniments generats> :==
$gen\_obj_1(eid:Z,atr_{11}:A_{11},...,atr_{1p}:A_{1p},time:T) \leftarrow objecte(O,T),W_1.$
:
:
$gen\_obj_n(eid:Z,atr_{n1}:A_{n1},...,atr_{ns}:A_{ns},time:T) \leftarrow objecte(O,T),W_n.$

Una regla de deducció d'esdeveniments generats defineix una classe d'esdeveniments generats associada a una classe d'objectes. Una de les classes d'esdeveniments, a la definició anterior és *gen_obj*$_i$, amb atributs *eid,atr*$_{i1}$,...,*atr*$_{ij}$,*time*. En aquesta definició, Z,A$_{i1}$,...,A$_{ip}$,T no són dominis, sinó que són les variables que prenen els valors dels diversos atributs. Cal notar que el domini dels atributs no apareix a la definició de la classe, sinó que es dedueix del cos de la regla de deducció en el cas dels atributs *atr*$_{i1}$,...,*atr*$_{ij}$ i és predefinit en el cas dels atributs *eid* i *time*.

Totes les classes d'esdeveniments generats tenen els atributs *eid* i *time*:

- El valor de l'atribut *eid* permet identificar els esdeveniments de la classe d'esdeveniments generats i és un valor assignat pel sistema en el moment en què ocorre l'esdeveniment. El seu domini és el dels identificadors interns dels esdeveniments de la classe *gen_obj*$_i$.
- El valor de l'atribut *time* correspon a l'instant de temps en què ocorre l'esdeveniment, i és un valor que es dedueix de la regla de deducció. El seu domini, *Time*, és la unitat de temps bàsica del sistema que es modela.

El cos d'una d'aquestes regles de deducció té sempre la forma: "← objecte(O,T),W$_i$.". Es tracta d'una conjunció entre el literal *objecte(O,T)* que fa referència a l'existència de l'objecte O, que és qui genera l'esdeveniment, a la classe *objecte* en l'instant en què aquest esdeveniment es genera, i una fórmula de primer ordre W$_i$ (més endavant veurem quins són els predicats que poden aparèixer en aquestes fórmules).

**Exemple 5.1.18**

En una classe d'objectes magatzem del sistema de gestió de vendes podria fer falta definir una classe d'esdeveniments generats que anunciïn que un magatzem ha fet una venda excel•lent, és a dir, una venda amb un import per sobre d'1 milió d'euros.

```
Extensional object class magatzem
  :
  generated events
  venda_excel.lent(eid:Z, art:A, mgt:M, temps:T) ←
                        magatzem(M,T),venda(W,T),
                        art(W:A),mgt(W:M),import(W:V),V > 1000000.
End
```

Els atributs de la classe d'esdeveniments generats són *eid, art, mgt* i *temps* que indiquen l'identificador intern de l'esdeveniment generat, l'article que s'ha venut, el magatzem i l'instant de temps en què s'ha fet la venda, respectivament. ♦

No es permet definir en un mateix MCDO dues classes d'esdeveniments generats amb un mateix nom.

**Restriccions d'integritat**

Les restriccions d'integritat corresponents a condicions que l'estat dels objectes d'una certa classe han de satisfer, es poden definir a la mateixa classe d'objectes:

**[Extensional | Intensional | Generalized] object class** objecte
:
[ **constraints** <definició restriccions d'integritat>]
:
**End**

Com en el cas de restriccions associades a una classe d'esdeveniments, les restriccions associades a una classe d'objectes són restriccions escrites en forma de denegació a les quals s'ha associat un predicat d'inconsistència per donar aparença de regla de deducció:

<definició restriccions d'integritat> ::= $incons\_obj_1(\mathbf{X_1}) \leftarrow objecte(O,T),W_1.$
:
$incons\_obj_n(\mathbf{X_n}) \leftarrow objecte(O,T),W_n.$

La seva forma és molt similar a la de les restriccions en classes d'esdeveniments. L'única diferència és que en la conjunció al cos de la restricció, hi apareix el literal *objecte(O,T)* en comptes del literal *esdev(Z,T)*. Més endavant veurem quins són els predicats que poden aparèixer a la fórmula de primer ordre, que és l'altre element en la conjunció.

No es permet definir en un mateix MCDO dues restriccions d'integritat amb un mateix predicat d'inconsistència.

**Exemple 5.1.19**

A la classe *article* dels exemples anteriors es pot definir una restricció d'integritat per evitar que hi pugui haver un article amb un preu superior a 50.000 euros:

**constraint** preuExcessiu(A,T) ← article(A,T),preu(A,T:Pr),Pr>50000. ♦

**Macros d'ajuda a l'escriptura de regles i restriccions**

Sovint al cos de les regles de deducció i en alguns casos al cos de les restriccions d'integritat, hi apareix una mateixa estructura:

$(esdev_1(Z_1,T1),T1 \leq T,W_1)$ ; ... ; $(esdev_i(Z_i,T1),T1 \leq T,W_i)$,
$not(esdev_j(Z_j,T2_j),W_j,T1 < T2_j,\ T2_j \leq T,...$
$...,esdev_n(Z_n,T2_n),W_n,T1 < T2_n,\ T2_n \leq T)$

Per simplificar l'escriptura d'aquestes regles, el llenguatge genèric per a MCDO que aquí introduïm ofereix dues macros: ***occurred*** i ***not_occurred***. Tot seguit s'exposa la correspondència entre les macros i la part de les regles que substitueixen.

**occurred**$(event_1(Z_1,T1),[W_1]$ ;...; $event_n(Z_n,T1),[W_n])$ ==
$((event_1(Z_1,T1),T1 \leq T,W_1)$ ; ... ; $(event_n(Z_n,T1),T1 \leq T,W_n)$.
**not_occurred**$((event_1(Z_1,T2_1),T1,[W_1]),$ ... , $(event_n(Z_n,T2_n),T1,[W_n])) ==$
$not(event_1(Z_1,T2_1),W_1,T1 < T2_1,\ T2_1 \leq T,...$
$...,event_n(Z_n,T2_n),W_n,T1 < T2_n,\ T2_n \leq T)$.

**Exemple 5.1.21**

Una regla en què aquestes macros són útils és el de la regla de deducció del valor de l'atribut *preuArt*, que sense utilitzar macros seria:

preuArt(A,T:Pr) ← ((**created** article(A,T1),T1≤T,preuInicial(A:Pr));
(nouPreu(Z,T1),T1≤T,art(Z:A),preu(Z:Pr))),
not(nouPreu(W,T2),T2>T1,T2≤T,art(W:A)).

i utilitzant les macros, és la següent:

preuArt(A,T:Pr) ← **ocurred**((**created** article(A,T1),[preuInicial(A:Pr)]);
(nouPreu(Z,T1),[art(Z:A),preu(Z:Pr)])),
**not_occurred**(nouPreu(W,T2),T1,[art(W:A)]). ♦

**Predicats a les regles de deducció i restriccions d'integritat**

Els predicats que poden aparèixer al cos de les regles i restriccions d'integritat en una classe d'objectes són el mateix tipus de predicats que poden aparèixer al cos de les regles i restriccions en una classe d'esdeveniments externs. És a dir, predicats avaluables, predicats auxiliars i predicats definits en el mateix MCDO.

**Exemple 5.1.20**

El valor de l'atribut *estocMgt* de la classe d'objectes *magatzem* defineix quin és l'estoc de cadascun dels articles del magatzem en un instant determinat:

```
Extensional object class magatzem
  :
  time varying attributes estocMgt: set of record of art: ARTICLE,
                                                      estoc: INTEGER.
  derivation rules
  estocMgt(M,T: [A, Q]) ← magatzem(M,T), article(A,T),
        sum(X,[occurred(venda(Z,T1),[art(Z:A),mgt(Z:M),qtt(Z:X)])],Qv),
        sum(Y,[occurred(noves_existencies(W,T1),
              [art(W:A),mgt(W:M),qtt(W:Y)])],Qn),
        Q=Qn-Qv.
End
```

Es tracta d'un atribut definit sobre un domini estructurat **set of record of**, és a dir, l'atribut és multivaluat i cadascun d'aquests valors indica un article que hi ha al magatzem i l'estoc d'aquest article. Al cos de la regla de deducció hi trobem gairebé tots els tipus de predicats de què hem parlat:

- Predicats avaluables. Ho és el del literal Q=Qn-Qv.
- Predicats auxiliars. En dues ocasions s'utilitza el predicat auxiliar *sum*.
- Predicats que fan referència a l'existència d'un objecte en una classe en el mateix instant per al qual es defineix el valor de l'atribut. Un d'aquests predicats apareix al literal article(A,T), que fa referència a l'existència de l'article A a la classe *article*.
- De predicats que facin referència al valor d'un atribut d'un objecte no n'hi ha cap. Ara bé, el literal estocMgt(M,T:[A,Q]) en el cos d'una regla, seria un exemple d'aquest tipus de predicats.
- Predicats que fan referència a un esdeveniment ocorregut en un instant anterior per al qual es defineix el valor de l'atribut. Dos d'aquests predicats apareixen als literals venda(Z,T1) i noves_existències(W,T1), que fan referència a l'ocurrència d'esdeveniments de les classes *venda* i *noves_existències*.
- Predicats que fan referència al valor d'un atribut d'un esdeveniment ocorregut en un instant anterior per al qual es defineix el valor de l'atribut. Tres d'aquests predicats apareixen als literals art(W:A),mgt(W:M) i qtt(W:Y) que fan referència al valor dels atributs *art*, *mgt* i *qtt* de l'esdeveniment W de la classe d'esdeveniments *noves_existències*. ♦

```
Event   nouTreballador (nom: STRING; dept:DEPARTAMENT). End
Event   baixaTreballador (treb: TREBALLADOR). End
Event   nouDepartament(). End
Event   canviDepartament (treb:TREBALLADOR; dept:DEPARTAMENT).
        constraint
        ésElDirector(E,T) ← treb(Z:E),dept(Z:D),responsable(D,T:E).
End
Event   nouSou (treb:TREBALLADOR; import:INTEGER). End
Event   nouDirector (dept:DEPARTAMENT; dir:TREBALLADOR).
        constraint
        jaÉsElDirector(E,T) ← dir(Z:E),dept(Z:D),responsable(D,T-1:E).
        noTreballaA(E,D,T) ← dir(Z:E),dept(Z:D),not(treballaA(E,T:D)).
End
```

**Figura 5.2.1** Classes d'esdeveniments externs

## 5.2 Sistema exemple

En aquesta secció presentem l'MCDO d'un sistema exemple que anirem utilitzant a la resta del capítol. La situació en què s'emmarca és una empresa que vol gestionar la situació dels empleats als seus departaments.

Els objectes més rellevants del domini que es vol modelar són els treballadors i els departaments. D'un treballador, es vol saber si és director d'algun departament, en quin departament treballa, qui és el seu cap i el sou que cobra. D'un departament, interessa conèixer qui és el seu responsable i els treballadors que té en plantilla. D'un treballador que és director de departament, es vol saber també el sobresou que cobra com a director, que és un tant per cent fix del sou de treballador.

No es vol permetre que: un treballador pugui canviar de departament, si és director del departament en què està; es pugui assignar a un treballador el càrrec de director d'un departament del qual ja ho és; es pugui assignar a un treballador el càrrec de director d'un departament en què no treballa. Per acabar, no es vol permetre que el nombre de persones de plantilla d'un departament sigui superior a 50.

El sistema, d'altra banda, ha d'avisar en el cas que algun treballador que no sigui director d'un departament passi a tenir un sou superior a 50.000 euros.

### Classes d'esdeveniments

A la figura 5.2.1 trobem la definició de les classes d'esdeveniments externs del sistema descrit. Conté les classes d'esdeveniments externs de creació i de destrucció

d'objectes de la classe *treballador*, que són *nouTreballador* i *baixaTreballador* i també la de creació d'objectes de la classe *departament*, *nouDepartament*. Hi ha, també, la classe d'esdeveniments externs per notificar el canvi de departament d'un treballador, *canviDepartament*. I, per acabar, hi ha la classe d'esdeveniments externs per notificar un canvi del sou d'un treballador, *nouSou*, i la classe d'esdeveniments externs, que notifica un canvi del director d'un departament, *nouDirector*.

S'ha especificat com a restricció d'integritat de la classe d'esdeveniments *canviDepartament* que no es pugui canviar de departament un treballador que és director del departament; i s'han especificat com a restriccions d'integritat de la classe d'esdeveniments *nouDirector*, que no es pugui assignar el càrrec de director d'un departament a un treballador que ja ho és i que no es pugui assignar el càrrec de director d'un departament a un treballador que no treballa al departament.

```
Extensional object class treballador
 created  treballador(oid:E, nom:N, deptInicial: D,time:T) ←
                              nouTreballador(Z,T), nom(Z:N), dept(Z:D).
 destroyed treballador(oid:E,time:T) ← baixaTreballador(Z,T), treb(Z:E).
 constant attributes          nom: String.
 time varying attributes  ésDirector: Departament.
                           treballaA: Departament.
                             cap: Director.
                            sou: Integer.
 derivation rules
 ésDirector(E,T:D) ← occurred(nouDirector(Z,T1),[treb(Z:E),dept(Z:D)]),
            not_occurred(nouDirector(W,T2),T1,[dept(W:D)]).
 treballaA(E,T:D) ← occurred((created treballador(E,T1),[deptInicial(E:D)]);
                    (canviDepartament(Z,T1),[treb(Z:E),dept(Z:D)])),
            not_occurred(canviDepartament(W,T2),T1,[treb(W:E)]).
 cap(E,T:M) ← treballaA(E,T:D), responsable(D,T:M).
 sou(E,T:S) ← occurred(nouSou(Z,T1),[treb(Z:E),import(Z:S)]),
            not_occurred(nouSou(W,T2),T1,[treb(W:E)]).
 generated events
 important(treb:E,total:S,time:T) ← nouSou(Z,T),treb(Z:E),import(Z:S),
                                    not(director(E,T)),S>50000.
End
```

**Figura 5.2.2** Classe d'objectes *treballador*

```
Extensional object class departament
  created departament(oid:D,time:T) ← nouDepartament(Z,T).
  time varying attributes    responsable: Director.
                             plantilla: set of Treballador.
  derivation rules
  responsable(D,T:M) ← occurred(nouDirector(Z,T1),[dept(Z:D),dir(Z:M)]),
                       not_occurred(nouDirector(W,T2), T1, [dept(W:D)]).
  plantilla(D,T:E) ← treballaA(E,T:D).
  constraints
  plantillaExcedida(D,N,T) ← count(E,plantilla(D,T:E),N),N>50.
End
```

**Figura 5.2.3** Classe d'objectes *departament*

**Classes d'objectes**

A la figura 5.2.2 apareix la definició de la classe d'objectes extensional *treballador*. Les classes d'esdeveniments de naixement i mort d'objectes a la classe són *createdTreballador* i *destroyedTreballador*. Els atributs d'un treballador són: *ésDirector*, que indica si el treballador és director del departament on treballa; *treballaA* que indica quin és el departament on treballa; *cap*, que indica qui és el treballador responsable del departament on treballa, i *sou*, que indica quants diners cobra anualment.

Observant les regles de deducció de cadascun dels atributs, es pot saber el seu valor en un cert instant:

- Un treballador E *ésDirector* del departament D en un cert instant T, si se li ha assignat el càrrec de director del departament D en un instant T1 anterior o igual a T i el departament D no ha canviat de director entre T1 i T.

- Un treballador E *treballaA* un departament D en un instant T, si l'han assignat com a treballador del departament D en un instant T1 anterior o igual a T i no l'han canviat de departament entre T1 i T.

- El *cap* d'un treballador E en un instant T és el responsable del departament on el treballador E treballa en l'instant T.

- Per acabar, el *sou* d'un treballador E en un instant T és S, si hi ha hagut un esdeveniment *nouSou* en un instant T1 anterior o igual a T que ha afectat el treballador i que li ha assignat un sou S i no hi ha hagut cap altre esdeveniment *nouSou* que hagi afectat el treballador entre T1 i T.

```
Intensional object class  director is a treballador
  derivation director(M,T) ← treballador(M,T), ésDirector(M,T).
  time varying attributes    sobreSou: Integer.
  derivation rules
   sobreSou(M,T:SS) ← sou(M,T:S), SS = S * 20/100.
End
```

**Figura 5.2.4** Classe d'objectes *director*

Per poder notificar el cas d'un treballador amb un sou superior a 50.000 euros s'ha definit una classe d'esdeveniments generats, *important*, a la classe d'objectes *treballador*.

A la figura 5.2.3 trobem la definició de la classe d'objectes extensional *departament*. Com que no hi ha definida una classe d'esdeveniments de mort, podem afirmar que els departaments que es creen a la classe *departament* mai no seran eliminats del sistema. Un departament té dos atributs: el *responsable* del departament i la *plantilla* del departament.

- El *responsable* d'un departament D en un instant T és M, si hi ha hagut un esdeveniment *nouDirector* en un instant T1 anterior o igual a T que ha afectat el departament D i que li ha assignat com a responsable M i no hi ha hagut cap altre esdeveniment *nouDirector* que hagi afectat el departament D entre T1 i T.
- Un treballador E és membre de la *plantilla* d'un departament D en un instant T si E treballa al departament D en el mateix instant T.

Per evitar que la plantilla d'un departament superi les 50 persones, s'ha definit una restricció d'integritat, *plantillaExcedida*, a la classe d'objectes *departament*.

Per acabar, en el model exemple hi ha definida la classe *director*, figura 5.2.4. Aquesta classe és una subclasse intensional de la classe *treballador*, i permet mantenir la informació sobre quin és el sobresou d'un director de departament:

- El *sobreSou* d'un director M en un instant T és el 20 per cent del *sou* de M com a *treballador* en el mateix instant.

## 5.3 Síntesi dels elements necessaris per determinar Esquemes d'Interacció

L'objectiu d'aquesta secció és explicar com fer la síntesi dels elements necessaris per a la determinació d'Esquemes d'Interacció a partir d'un MCDO. Aquesta explicació la donarem element per element en dues subseccions. A la primera subsecció explicarem

com fer la síntesi dels elements bàsics, i a la segona subsecció com fer la síntesi de les regles d'esdeveniments. Tant en una secció com a l'altra donarem exemples basats en el model exemple de la secció 5.2. La síntesi completa del model exemple es pot trobar a l'annex 1 que s'inclou al final del capítol.

### 5.3.1 Síntesi d'elements bàsics

Els elements bàsics són els elements relacionats amb classes d'esdeveniments externs, classes d'objectes i classes d'esdeveniments estructurals.

**Classes d'esdeveniments externs**

En la síntesi ens interessa tenir el conjunt de noms de classes d'esdeveniments externs definides en un model. Aquest conjunt es coneix amb el nom d'*EsdExterns*.

En els MCDO, els noms d'aquestes classes d'esdeveniments són els noms de les classes d'esdeveniments externs al model.

**Exemple 5.3.1**

Al sistema exemple, el conjunt de noms de les classes d'esdeveniments externs, figura 5.2.1, és EsdExterns = {nouTreballador, baixaTreballador, nouDepartament, canviDepartament, nouSou, nouDirector}. ♦

**Classes d'objectes**

En la síntesi necessitem el conjunt de noms de les classes d'objectes definides en un model. Aquest conjunt es coneix amb el nom d'*Objectes*.

En els MCDO, els noms d'aquestes classes d'objectes són els noms de les classes d'objectes i els noms de les classes d'esdeveniments externs al model. El motiu pel qual considerem els esdeveniments externs com a objectes és que hi ha definits atributs derivats i restriccions d'integritat en què, per ser avaluats, cal conèixer l'estat del sistema en el moment en què ocorren els esdeveniments. Això vol dir que els canvis en l'estat dels objectes en el mateix moment en què ocorren uns esdeveniments determinats pot afectar el valor dels seus atributs derivats o pot provocar la violació de les seves restriccions d'integritat. Així, als Esquemes d'Interacció faran falta, en aquests casos, tipus d'interacció per enviar els esdeveniments estructurals que reflecteixin aquests canvis d'estat a les classes d'esdeveniments externs que es poden veure afectades.

**Exemple 5.3.2**

Al sistema exemple, figures 5.2.1,...,5.2.4, el conjunt de noms de les classes d'objectes és Objectes = {treballador, departament, director, nouTreballador, baixaTreballador, nouDepartament, canviDepartament, nouSou, nouDirector}. ♦

**Atributs.** En la síntesi interessa el conjunt de noms d'atributs de cadascuna de les classes d'objectes en un model. Aquest conjunt es coneix amb el nom d'*Atributs(CO)*, on CO és la classe d'objectes.

En els MCDO els atributs d'una classe d'objectes o bé estan definits en la mateixa classe o bé s'hereten d'alguna de les seves superclasses.

D'altra banda, els atributs d'una classe d'objectes definida en l'MCDO com una classe d'esdeveniments externs, són els atributs definits a la classe d'esdeveniments com a derivats. La resta, és a dir, els atributs que prenen el seu valor de l'entorn, no són necessaris en la síntesi, perquè el seu valor en cap cas no depèn de l'estat del sistema.

**Exemple 5.3.3**

La classe d'objectes *treballador* té definits cinc atributs: *nom, ésDirector, treballaA, cap* i *sou*. Així, el conjunt de noms d'atributs de la classe d'objectes *treballador* és Atributs(treballador) = {treballadorNom, treballadorÉsDirector, treballadorTreballaA, treballadorCap, treballadorSou}. En ser atributs heretats per la classe *director*, posem com a prefix en el nom de l'atribut, el nom de la classe d'objectes *treballador*.

La classe d'objectes *director* té un atribut definit a la seva pròpia classe i cinc atributs heretats de les seves superclasses. Així, el conjunt de noms d'atributs de la classe *director* és Atributs(director) = {sobreSou, directorNom, directorÉsDirector, directorTreballaA, directorCap, directorSou}.

Al sistema exemple no hi ha cap classe d'esdeveniments externs amb atributs derivats. Per aquest motiu, podem dir que des del punt de vista de la síntesi cap de les classes d'objectes corresponents a classes d'esdeveniments externs tenen atributs. ♦

**Esdeveniments generats.** En la síntesi ens interessa el conjunt de noms de les classes d'esdeveniments generats per a cada classe d'objectes d'un model. Aquest conjunt rep el nom de *Generacions(CO)*, on CO és la classe d'objectes.

En els MCDO, aquestes classes d'esdeveniments són les classes d'esdeveniments generats, les classes d'esdeveniments de naixement i les classes d'esdeveniments de mort definides a la classe d'objectes.

**Exemple 5.3.4**

A la classe d'objectes *treballador* del sistema exemple, figura 5.2.3, hi ha definida la classe d'esdeveniments generats *important*, la classe d'esdeveniments de naixement *createdTreballador* i la classe d'esdeveniments de mort *destroyedTreballador*. Així, Generacions(treballador) = {important, createdTreballador, destroyedTreballador}. ♦

**Restriccions d'integritat.** En la síntesi ens interessa tenir el conjunt de predicats d'inconsistència corresponents a les restriccions d'integritat definides per a cada classe d'objectes d'un model. Aquest conjunt es coneix amb el nom de *Restriccions(CO)*, on CO és la classe d'objectes.

En els MCDO, les restriccions es troben definides a les classes d'objectes. Cada restricció té associat un predicat d'inconsistència que s'utilitzarà en la síntesi per identificar-la.

**Exemple 5.3.5**

Al sistema exemple, la classe d'objectes *departament* té definida la restricció d'integritat *plantillaExcedida*, figura 5.2.2,. Així, Restriccions(departament) = {plantillaExcedida}.

Un cop explicitades les restriccions d'integritat implícites a la classe d'esdeveniments externs *nouTreballador*, arribem a una única restricció d'integritat, que anomenarem *ic1*:

ic1(E,T) ← nouTreballador(Z,T),
treb(Z:E),dept(Z:D),not(departament(D,T)).

Aquesta restricció indica que no és possible que en un instant T donem d'alta un nou treballador i l'assignem a un departament que no existeix en aquest instant. Així, Restriccions(nouTreballador) ={ic1}. ♦

**Classes d'esdeveniments estructurals**

En la síntesi ens interessen els tipus d'esdeveniment estructurals següents:

- Esdeveniments estructurals d'inserció d'objectes en una classe. Inserció(CO) és el nom de la classe d'esdeveniments estructurals d'inserció d'objectes de la classe CO.
- Esdeveniments estructurals d'esborrat d'objectes d'una classe. Esborrat(CO) és el nom de la classe d'esdeveniments estructurals d'esborrat d'objectes de la classe CO.
- Esdeveniments estructurals de modificació del valor d'un atribut dels objectes d'una classe d'objectes. El conjunt de noms de les classes d'esdeveniments estructurals de modificació del valor d'un atribut A és Modificació(A).

- Esdeveniments estructurals de violació de restriccions d'integritat. El nom de la classe d'esdeveniments estructurals de violació d'una restricció d'integritat Ic es Restricció(Ic).

- Esdeveniments estructurals de generació d'esdeveniments. El nom de la classe d'esdeveniments estructurals de generació d'esdeveniments d'una classe G es designa Generació(G).

Ara veurem quines són les classes d'esdeveniments estructurals per a una classe d'objectes en un MCDO:

- La classe d'esdeveniments estructurals d'inserció d'objectes en una classe d'objectes no es defineix explícitament en aquests models, i ens hi referirem com *i_objecte*. Cal notar que *createdObjecte* no representa aquesta classe d'esdeveniments estructurals, perquè *createdObjecte* és una classe de generació d'esdeveniments. És com a resultat de la generació d'un esdeveniment de la classe *createdObjecte* que es provocarà l'operació primitiva d'inserció d'un objecte *i_objecte*. Per acabar, només cal dir que les classes d'objectes corresponents a classes d'esdeveniments externs no tenen aquest tipus d'esdeveniments estructurals.

- La classe d'esdeveniments estructurals d'esborrat d'objectes d'una classe tampoc no es defineix explícitament en aquests models, i ens hi referirem com *d_objecte*. Pel mateix motiu podem dir que *destroyedObjecte* no representa aquesta classe d'esdeveniments estructurals. Les classes d'objectes corresponents a classes d'esdeveniments externs tampoc no tenen aquest tipus d'esdeveniments estructurals.

- Les classes d'esdeveniments estructurals de modificació dels valors dels atributs no estan mai definides explícitament en els MCDO. En general, tots els atributs tenen dues classes d'esdeveniments estructurals de modificació, una d'inserció *i_nomAtribut* i una d'esborrat *d_nomAtribut*. Una modificació d'un atribut univaluat correspon a dos esdeveniments estructurals un d'inserció i un d'esborrat en un mateix instant de temps, on s'entén que primer s'aplica l'esdeveniment d'esborrat i després el d'inserció. En el cas dels atributs multivaluats, una inserció representa afegir un valor més a l'atribut, i un esborrat correspon a eliminar un valor dels que té l'atribut.

- Per la relació biunívoca existent entre una restricció d'integritat i la corresponent classe d'esdeveniments estructurals de violació, podríem utilitzar el mateix nom per a la classe d'esdeveniments estructurals de violació que per al predicat d'inconsistència que fa referència a la restricció. De totes maneres, no ho farem per motius que vénen donats per la manera de fer la síntesi de regles d'esdeveniments, i anomenarem la classe

d'esdeveniments estructurals *i_restricció* (on *restricció* és el nom del predicat d'inconsistència).

- Per la relació biunívoca existent entre una classe d'esdeveniments generats i la corresponent classe d'esdeveniments estructurals de generació, podríem utilitzar el mateix nom per a ambdues. De totes maneres, no ho farem per motius que vénen donats per la manera de fer la síntesi de regles d'esdeveniments, i anomenarem la classe d'esdeveniments estructurals *i_esdGenerat* (on *esdGenerat* és el nom de la classe d'esdeveniments generats).

**Exemple 5.3.6**

Les classes d'esdeveniments estructurals per a la modificació de l'estat d'objectes de la classe *treballador* són les següents. Cal tenir en compte que aquesta classe no té definida cap restricció d'integritat.

Inserció(treballador) = i_treballador.
Esborrat(treballador) = d_treballador.
Modificació(treballadorNom) = {i_treballadorNom,d_treballadorNom}
Modificació(treballadorÉsDirector) = {i_treballadorÉsDirector,
d_treballadorÉsDirector}
Modificació(treballadorTreballaA) = {i_treballadorTreballaA,
d_treballadorTreballaA}
Modificació(treballadorCap) = {i_treballadorCap, d_treballadorCap}
Modificació(treballadorSou) = {i_treballadorSou, d_treballadorSou}
Generació(important) = {i_important} ♦

## 5.3.2 Síntesi de regles d'esdeveniments

La síntesi de regles d'esdeveniments no la farem a partir del mateix MCDO, sinó a partir de les regles de transició i d'esdeveniments interns que es poden obtenir a partir de l'MCDO. El procediment d'obtenció d'aquestes regles per a models deductius es pot trobar a [San94], i pot ser aplicat a MCDO després de seguir tres passos.

Primerament: fer explícites totes les regles i parts de regles implícites en l'MCDO, i transformar a forma normal (a regles el cos de les quals és una conjunció de literals, sent cada literal o bé un àtom o un àtom negat) totes les regles en el model que no hi estiguin. La transformació d'una regla a forma normal es pot fer seguint el que descriu [Llo87].

**Exemple 5.3.7**

Al sistema exemple, presentat a la secció 5.2 d'aquest capítol, hi ha únicament quatre regles que no estan en forma normal. Es tracta de les regles que defineixen el valor dels atributs *responsable*, *ésDirector*, *treballaA* i *sou*. Agafem el cas de la regla de definició de l'atribut *treballaA* un cop desfetes les macros de simplificació (occurred, not occurred):

treballaA(E,T:D) ← ((**created** treballador(E,T1),T1≤T,deptInicial(E:D)) ;
(canviDepartament(Z,T1),T1≤T,treb(Z:E),dept(Z:D)),
not(canviDepartament(W,T2),T2>T,T2≤T1,treb(W:E)).

Aquesta regla presenta dues característiques que fan que no sigui una regla en forma normal. La primera és que al cos de la regla hi ha un literal negat no atòmic. La segona és que al cos de la regla hi ha una disjunció. Per eliminar el literal no atòmic cal reescriure la regla fent servir un predicat auxiliar, que aquí anomenarem *canviDept*. Per eliminar la disjunció, cal desglossar en dues regles la definició del predicat *treballaA*. El resultat és el següent:

treballaA(E,T:D) ← **created**Treballador(E,T1),T1≤T,
deptInicial(E:D),not(canviDept(E,T1,T)).
treballaA(E,T:D) ← canviDepartament(Z,T1),T1≤T,
treb(Z:E),dept(Z:D),not(canviDept(E,T1,T)).
canviDept(E,T1,T) ← canviDepartament(Z,T2),T1<T2,T2≤T,treb(Z:E). ♦

El segon pas que cal fer és: agafar les condicions en els predicats auxiliars predefinits que no corresponguin a un únic literal positiu i substituir-les per predicats auxiliars, definir una nova regla per a cadascun d'aquests predicats auxiliars, el cos de la qual sigui la condició que han substituït, i transformar les noves regles a forma normal.

**Exemple 5.3.8**

Al sistema exemple només trobem una regla amb un literal de predicat auxiliar predefinit, es tracta de la regla que defineix la restricció d'integritat *plantillaExcedida*. Com que la condició en aquest predicat és un únic literal positiu, no cal fer cap canvi en aquesta regla abans de passar a l'obtenció de les regles de transició i d'esdeveniments interns. ♦

Per últim, el tercer pas és: classificar els predicats al model segons la seva *dimensió positiva*. La dimensió d'un predicat caracteritza la manera en què la seva extensió evoluciona en el temps. Se'n poden distingir dues dimensions: la positiva i la negativa (no cal per al procediment d'obtenció de regles de transició i d'esdeveniments interns).

La definició de *dimensió positiva* d'un predicat que podem trobar a [San94], figura a l'annex 2 d'aquest capítol, com també la caracterització de la dimensió positiva dels predicats segons el paper que fan en un MCDO.

El procediment d'obtenció de les regles de transició i d'esdeveniments interns [San94], en aquesta tesi no el reproduirem, simplement direm que es basa en la transformació d'un subconjunt dels literals que es poden trobar al cos de les regles. Concretament, els literals que es transformen són aquells literals de predicats que apareixen al cos de les regles i que estan definits sobre l'instant de temps actual T. Els literals de predicats avaluables i els literals de predicats que fan referència al valor d'atributs d'esdeveniments no es transformen perquè no estan definits sobre un instant de temps. Tampoc no es transformen els literals de predicats definits per a un instant de temps T1, essent T1 anterior a l'instant actual T.

Atès que la transformació de literals de predicats auxiliars predefinits no està resolta a [San94], els deixarem sense transformar en les regles resultants. Aquesta circumstància no afecta, de totes maneres, el procés de síntesi de regles d'esdeveniments que explicarem ara.

La taula de transformació de literals en un MCDO la podem trobar també a l'annex 2 d'aquest capítol; es tracta, simplement, d'una adaptació de la que dóna [San94] per a literals en models deductius.

**Regles de transició i d'esdeveniments interns**

Les regles de transició i d'esdeveniments interns ens proporcionaran, per a cada tipus d'esdeveniment intern, un conjunt de regles que definiran què és el que pot fer que un esdeveniment del tipus ocorri en un cert instant de temps T. Concretament, ens diran quins esdeveniments externs i/o interns han d'ocórrer en el mateix instant T i quin havia de ser l'estat del sistema en els instants anteriors a T.

El que en aquestes regles s'anomena esdeveniments interns són el que en la síntesi anomenem esdeveniments estructurals, és a dir, operacions bàsiques sobre l'estat d'un sistema.

Així, les regles d'esdeveniments interns/estructurals defineixen quan es produirà un esdeveniment intern/estructural d'un tipus/classe en un instant de temps T en funció de l'estat del sistema en els instants T-1 i T, concretament en funció de la part de l'estat del sistema que aquest esdeveniment modifica. D'altra banda, les regles de transició defineixen quin és l'estat del sistema en un instant T en funció de l'estat del sistema en

instants anteriors a T i els esdeveniments externs i/o interns/estructurals que puguin ocórrer en T.

**Exemple 5.3.9**

A la figura 5.3.1 es poden veure les regles de transició i d'esdeveniments interns corresponents als tipus d'esdeveniments interns (classes d'esdeveniments estructurals) *i_plantilla* i *d_plantilla* per a la modificació de l'atribut *plantilla* del sistema exemple. ♦

**Exemple 5.3.10**

A la figura 5.3.2, hi trobem les regles de transició i d'esdeveniments interns per a la restricció d'integritat *plantillaExcedida*. Cal recordar que els literals de predicats auxiliars predefinits no es transformen, per aquest motiu es pot veure el predicat auxiliar predefinit *count* amb la condició plantilla(D,T,N) tal com apareixia en l'MCDO. ♦

El procediment de generació de regles de transició i d'esdeveniments interns presentat a [San94] proposa diverses estratègies per simplificar les regles de transició i d'esdeveniments interns. Malgrat això, aquesta simplificació no sempre pot ser tan considerable com es voldria, i, per tant, ens interessa que la síntesi de regles d'esdeveniments es pugui fer tant a partir de regles de transició i d'esdeveniments interns simplificades com sense simplificar.

**Regles d'esdeveniments interns**

**I.1** i_plantilla(D,T:E) ← i_plantilla$_1$(D,T:E).

**I.2** i_plantilla$_1$(D,T:E) ← plantilla$_{12}$(D,T:E),not(plantilla$_1$(D,T-1:E)).
**I.3** i_plantilla$_1$(D,T:E) ← plantilla$_{13}$(D,T:E),not(plantilla$_1$(D,T-1:E)).
**I.4** i_plantilla$_1$(D,T:E) ← plantilla$_{14}$(D,T:E),not(plantilla$_1$(D,T-1:E)).

**I.5** d_plantilla(D,T:E) ← d_plantilla$_1$(D,T:E)

**I.6** d_plantilla$_1$(D,T:E) ← plantilla$_1$(D,T-1:E),
not(plantilla$_{11}$(D,T:E)),not(plantilla$_{12}$(D,T:E)),
not(plantilla$_{13}$(D,T:E)),not(plantilla$_{14}$(D,T:E)).

**Regles de transició**

**T.1** plantilla(D,T:E) ← plantilla$_1$(D,T:E).

**T.2** plantilla$_1$(D,T:E) ← plantilla$_{11}$(D,T:E).
**T.3** plantilla$_1$(D,T:E) ← plantilla$_{12}$(D,T:E).
**T.4** plantilla$_1$(D,T:E) ← plantilla$_{13}$(D,T:E).
**T.5** plantilla$_1$(D,T:E) ← plantilla$_{14}$(D,T:E).

**T.6** plantilla$_{11}$(D,T:E) ← departament(D,T-1),treballaA(E,T-1:D),not(d_treballaA(E,T:D)).
**T.7** plantilla$_{12}$(D,T:E) ← departament(D,T-1), i_treballaA(E,T:D).
**T.8** plantilla$_{13}$(D,T:E) ← i_departament(D,T),treballaA(E,T-1:D),not(d_treballaA(E,T:D)).
**T.9** plantilla$_{14}$(D,T:E) ← i_departament(D,T),i_treballaA(E,T:D).

**Figura 5.3.1** Regles *plantilla*

**Regles d'esdeveniments interns**

I.1 i_plantillaExcedida(D,N,T) ← i_plantillaExcedida$_1$(D,N,T).

I.2 i_plantillaExcedida$_1$(D,N,T) ← plantillaExcedida$_{11}$(D,T:E),not(plantillaExcedida$_1$(D,T-1:E)).

**Regles de transició**

T.1 plantillaExcedida(D,N,T) ← plantillaExcedida$_1$(D,N,T).

T.2 plantillaExcedida$_1$(D,N,T) ← plantillaExcedida$_{11}$(D,N,T)

T.3 plantillaExcedida$_{11}$(D,N,T) ← count(E,plantilla(D,N,T),N).

**Figura 5.3.2** Predicat *plantillaExcedida*

**Regles d'esdeveniments interns simplificades**

I.1 i_plantilla(D,T:E) ← i_treballaA(E,T:D).

I.2 d_plantilla(D,T:E) ← d_treballaA(E,T:D),treballaA(E,T-1:D).

**Figura 5.3.3** Predicat *plantilla*

**Exemple 5.3.11**

A la figura 5.3.3 hi ha les regles de transició i d'esdeveniments interns de *plantilla* després d'haver passat pel procés de simplificació. ♦

**Síntesi de regles d'esdeveniments a partir de regles de transició i d'esdeveniments interns**

Abans de res cal tenir clar que el nostre objectiu és arribar a obtenir les regles d'esdeveniments que es refereixen a cadascuna de les classes d'esdeveniments estructurals per a un MCDO. És a dir, arribar a obtenir els conjunts de classes d'esdeveniments causa de cadascuna de les classes d'esdeveniments estructurals i els corresponents conjunts de classes d'esdeveniments prerequisit.

**Obtenció d'informació rellevant per fer la síntesi.** Les regles de transició i d'esdeveniments interns/estructurals tenen molta informació que no interessa a l'hora de fer la síntesi de regles d'esdeveniments. Tot seguit farem un estudi de cada conjunt d'aquestes regles, corresponents a un cert predicat en l'MCDO, que tindrà com a fi eliminar aquella informació que no ens cal. Concretament, eliminarem:

- Els literals antics de predicats definits sobre un instant anterior a l'instant actual T. El motiu és que nosaltres el que volem és trobar les causes que ocorrin esdeveniments d'una classe d'esdeveniments estructurals en un cert instant T. Com que el que provoca això són esdeveniments externs ocorreguts en T (cal recordar que no considerem models amb comportament "espontani"), no ens interessen en aquest estudi els literals antics al cos de les regles.

- Els literals avaluables. El què un literal avaluable al cos d'una regla sigui cert o fals no depèn de l'ocurrència d'uns certs esdeveniments, sinó dels valors de les variables en aquest literal. Està clar que perquè les variables tinguin uns certs valors, han d'ocórrer uns esdeveniments que instanciïn aquests valors o el sistema ha de tenir un estat determinat. Si el fet que tinguin aquests valors depèn de que ocorrin uns certs esdeveniments, la necessitat que ocorrin ja vindrà representada, a la regla, pel predicat corresponent, que farà referència a l'ocurrència d'aquests esdeveniments.

- Els literals de predicats que fan referència al valor de l'atribut d'un esdeveniment. Està clar que perquè un d'aquests predicats sigui cert, ha d'ocórrer un esdeveniment de la classe per a la qual estan definits. Ara bé, la necessitat que ocorri aquest esdeveniment ja vindrà representada, a la regla, pel predicat corresponent, que farà referència a l'ocurrència de l'esdeveniment.

- Les variables als predicats de cada literal al cos de les regles i als àtoms del cap de les regles. A les regles d'esdeveniments que volem obtenir només hi ha les classes d'esdeveniments causa i prerequisit de que es produeixin o no esdeveniments d'una classe d'esdeveniments estructurals, però en cap moment es donen condicions sobre els valors dels atributs d'aquests esdeveniments.

**I.1** i_plantilla ← i_plantilla$_1$.

**I.2** i_plantilla$_1$ ← plantilla$_{12}$.
**I.3** i_plantilla$_1$ ← plantilla$_{13}$.
**I.4** i_plantilla$_1$ ← plantilla$_{14}$.

**I.5** d_plantilla ← d_plantilla$_1$.

**I.6** d_plantilla$_1$ ← not(plantilla$_{11}$),not(plantilla$_{12}$),
not(plantilla$_{13}$),not(plantilla$_{14}$).

**T.1** plantilla ← plantilla$_1$.

**T.2** plantilla$_1$ ← plantilla$_{11}$.
**T.3** plantilla$_1$ ← plantilla$_{12}$.
**T.4** plantilla$_1$ ← plantilla$_{13}$.
**T.5** plantilla$_1$ ← plantilla$_{14}$.

**T.6** plantilla$_{11}$ ← not(d_treballaA).
**T.7** plantilla$_{12}$ ← i_treballaA.
**T.8** plantilla$_{13}$ ← i_departament,not(d_treballaA).
**T.9** plantilla$_{14}$ ← i_departament,i_treballaA.

**Figura 5.3.4** Informació rellevant predicat *plantilla*

**Regles d'esdeveniments interns**
**I.1** i_plantillaExcedida ← i_plantillaExcedida$_1$.

**I.2** i_plantillaExcedida$_1$ ← plantillaExcedida$_{11}$.

**Regles de transició**
**T.1** plantillaExcedida ← plantillaExcedida$_1$.

**T.2** plantillaExcedida$_1$ ← plantillaExcedida$_{11}$.

**T.3** plantillaExcedida$_{11}$ ← aux(plantilla).

**Figura 5.3.5** Informació rellevant predicat *plantillaExcedida*

Així, després d'eliminar parts no rellevants ens queden regles on el cos és una conjunció de literals que poden ser positius o negatius. En el cas de literals de predicats auxiliars predefinits en què la condició és un predicat definit sobre l'instant actual T, a les regles resultants es manté el literal, s'explicita el predicat en la condició i s'indica que es tracta d'un predicat que apareixia en un literal d'un predicat auxiliar predefinit, "aux".

**Exemple 5.3.12**

A la figura 5.3.4 trobem la informació que ens interessa de les regles de transició i d'esdeveniments interns del predicat *plantilla*, figura 5.3.1, i que utilitzarem per fer la síntesi de regles d'esdeveniments que es referiran a les classes d'esdeveniments estructurals *i_plantilla* i *d_plantilla* . ♦

**Exemple 5.3.13**

A la figura 5.3.5 apareix la informació que ens interessa de les regles de transició i d'esdeveniments interns del predicat *plantillaExcedida*, figura 5.3.3, i que utilitzarem per fer la síntesi de regles d'esdeveniments que es referiran a la classe d'esdeveniments estructurals *i_plantillaExcedida*. ♦

**Processament de la informació rellevant.** A partir de les regles de transició i d'esdeveniments interns en què només queda ja la informació rellevant per a la síntesi de regles d'esdeveniments, cal aplicar els passos següents:

- Agafar les regles que defineixen la classe d'esdeveniments estructurals en què estem interessats.
- Desplegar els predicats al cos de cadascuna de les regles a partir de la seva definició en altres regles del conjunt. Aplicar aquest pas tantes vegades com calgui fins a arribar a regles en què tots els literals són de predicats corresponents a classes d'esdeveniments

externs/estructurals o de predicats auxiliars predefinits. El cos de les regles resultants serà una fórmula de primer ordre.

- Passar el cos de cadascuna de les regles a forma normal disjuntiva.

**Exemple 5.3.14**

En el cas de les classes d'esdeveniments estructurals *i_plantilla* i *d_plantilla*, el resultat dels passos anteriors és:

i_plantilla ← i_treballaA ;
(i_departament,not(d_treballaA)) ;
(i_departament,i_treballaA).
d_plantilla ← d_treballaA,not(i_treballaA),not(i_departament) ;
d_treballaA,not(i_treballaA). ♦

**Determinació de solucions.** A partir de cadascuna de les conjuncions en una de les regles resultants del processament de la informació rellevant, podem deduir un o més conjunts de classes d'esdeveniments causa i prerequisit de l'ocurrència d'esdeveniments estructurals de la classe en el cap de les regles. Per trobar els conjunts de classes d'esdeveniments causa, ens cal trobar les classes d'esdeveniments externs/estructurals que han d'ocórrer per fer certa la conjunció. Per trobar els conjunts de classes d'esdeveniments prerequisit, ens cal trobar les classes d'esdeveniments externs/estructurals que no han d'ocórrer per fer certa la conjunció, o bé que la seva ocurrència pot afectar perquè la conjunció sigui certa o falsa.

Si en una conjunció no hi ha cap literal de predicat auxiliar predefinit, és a dir només hi ha literals positius i negatius de classes d'esdeveniments externs/estructurals, llavors, l'única manera de fer certa la conjunció és que ocorrin esdeveniments de cadascuna de les classes d'esdeveniments als literals positius de la conjunció. Les classes d'esdeveniments en aquests literals formaran l'únic conjunt de classes d'esdeveniments causa que es podrà deduir de la conjunció. D'altra banda, per fer certa la conjunció no hauran d'ocórrer esdeveniments de les classes d'esdeveniments als literals negatius de la conjunció. Així, les classes d'esdeveniments en aquests literals formaran part del corresponent conjunt de classes d'esdeveniments prerequisit.

**Exemple 5.3.15**

Això és el que passa a les conjuncions de les regles a què s'arriba per a les classes d'esdeveniments estructurals *i_plantilla* i *d_plantilla*, exemple 5.3.14. Agafem per exemple el cas de la segona conjunció en la regla de *i_plantilla*, *(i_departament,not(d_treballaA))*. Per fer certa aquesta conjunció, cal que *i_departament* sigui cert i que *d_treballaA* sigui fals. Tenint en compte el que representen

*i_departament* i *d_treballaA*, podem dir que cal que ocorrin esdeveniments de la classe *i_departament* i que no n'ocorrin de la classe *d_treballaA*. Per tant, *i_departament* formarà un conjunt de classes d'esdeveniments causa i *d_treballaA* formarà el corresponent conjunt de classes d'esdeveniments prerequisit.

Tot seguit es donen tots els conjunts de classes d'esdeveniments causa i prerequisit necessaris per fer certes les conjuncions de la regla *i_plantilla* i de la regla *d_plantilla*:

| | causa | prerequisit |
|---|---|---|
| i_plantilla | {i_treballaA} | { } |
| | {i_departament} | {d_treballaA} |
| | {i_departament, i_treballaA} | { } |

| | causa | prerequisit |
|---|---|---|
| d_plantilla | {d_treballaA} | {i_departament,i_treballaA} |
| | {d_treballaA} | {i_treballaA} ♦ |

Si en una conjunció hi ha un o més literals de predicats auxiliars predefinits, i també hi ha un o més literals positius de classe d'esdeveniments externs/estructurals, per fer certa la conjunció és essencial que ocorrin esdeveniments de cadascuna de les classes d'esdeveniments als literals positius de la conjunció. Les classes d'esdeveniments en aquests literals formaran l'únic conjunt de classes d'esdeveniments causa que es podrà deduir de la conjunció. D'altra banda, per fer certa la conjunció no hauran d'ocórrer esdeveniments de les classes d'esdeveniments als literals negatius de la conjunció.

El fet que ocorrin o no esdeveniments estructurals corresponents als predicats als literals de predicats auxiliars predefinits de la conjunció no és mai definitiu per fer certa o falsa la conjunció, ja que el fet que sigui certa o falsa dependrà del valor que s'acabi calculant mitjançant els predicats auxiliars predefinits. Així, seran les classes d'esdeveniments als literals negatius i les classes d'esdeveniments estructurals corresponents als predicats als literals de predicats auxiliars predefinits, les que formaran part del conjunt de classes d'esdeveniments prerequisit.

**Exemple 5.3.16**

En el model exemple no trobem cap cas d'aquests. Ara bé, suposem la conjunció següent: *(i_objecte,aux(atribut1),not(d_atribut2))*. Per fer certa aquesta conjunció cal que *i_objecte* sigui cert i que *d_atribut2* sigui fals. Que sigui cert o no *atribut1* no vol dir res, ja que el que afectarà serà el valor que es calculi mitjançant el predicat auxiliar predefinit en què estigui. D'altra banda, tenint en compte el que representen *i_objecte* i *i_atribut2*, podem dir que cal que ocorrin esdeveniments de la classe *i_objecte* i que no

n'ocorrin de la classe *i_atribut2*. Per tant, *i_objecte* formarà un conjunt de classes d'esdeveniments causa i *i_atribut2*, *i_atribut1*, *d_atribut1* (classes d'esdeveniments estructurals corresponents al predicat *atribut1*) formaran part del conjunt de classes d'esdeveniments prerequisit. ♦

Finalment, considerem el cas d'una conjunció en què hi ha un o més literals de predicats auxiliars predefinits, però no hi ha cap literal positiu de classe d'esdeveniments externs/estructurals. Ja hem comentat abans que el fet que ocorrin o no esdeveniments estructurals corresponents al predicat als literals de predicats auxiliars predefinits no és mai definitiu per fer certa o falsa la conjunció, ja que dependrà del valor que s'acabi calculant mitjançant els predicats auxiliars predefinits. Ara bé, tenint en compte que els models amb què tractem no tenen comportament "espontani", si la conjunció no inclou cap literal positiu de classe d'esdeveniments estructurals, per fer certa la conjunció haurà d'ocórrer com a mínim un esdeveniment extern/estructural d'una de les classes d'esdeveniments estructurals corresponents als predicats en literals de predicats auxiliars predefinits.

Així, si la conjunció té dos literals de predicats auxiliars predefinits, i als predicats de cadascun dels literals els corresponen dues classes d'esdeveniments estructurals, podem deduir que hi haurà quatre (2+2) conjunts de classes d'esdeveniments causa, formats cadascun per una de les classes d'esdeveniments estructurals. El conjunt de classes d'esdeveniments prerequisit per a cada conjunt de classes d'esdeveniments causa estarà format pels noms de les classes d'esdeveniments en literals negatius de la conjunció i per aquelles classes d'esdeveniments estructurals corresponents a predicats de literals de predicats auxiliars predefinits que no estiguin al conjunt corresponent de classes d'esdeveniments causa.

**Exemple 5.3.17**

En el model exemple no trobem cap cas representatiu d'aquests. Ara bé, suposem la conjunció següent: *(aux(atribut1),aux(objecte) not(d_atribut2))*. Que sigui certa o no la conjunció dependrà dels valors que es calculin mitjançant els predicats auxiliars predefinits. Ara bé, per provocar algun canvi en un model, sabem que ha d'haver ocorregut algun esdeveniment; per tant, perquè sigui certa la conjunció haurà d'haver ocorregut com a mínim un esdeveniment de les classes d'esdeveniments estructurals corresponents a *atribut1* i *objecte*. D'altra banda, tenint en compte el que representa *d_atribut2*, podem dir també que cal que no ocorrin esdeveniments de la classe *d_atribut2*. Per tant, com que les classes d'esdeveniments estructurals corresponents al predicat *atribut1* són *i_atribut1*, *d_atribut1* i hi ha una única classe d'esdeveniments estructurals corresponent al predicat *objecte*, que és *i_objecte*, podem dir que hi haurà

tres conjunts de classes d'esdeveniments causa i els corresponents conjunts de classes d'esdeveniments prerequisit, que seran:

| causa | prerequisit |
|---|---|
| {i_atribut1} | {i_atribut2,i_objecte,d_atribut1} |
| {d_atribut1} | {i_atribut2,i_objecte,i_atribut1} |
| {i_objecte} | {i_atribut2,d_atribut1,i_atribut1} ♦ |

**Exemple 5.3.18**

En el model exemple trobem un cas en què la informació rellevant extreta de regles de transició i d'esdeveniments interns conté un literal de predicat auxiliar predefinit, figura 5.3.5. Es tracta de la informació rellevant per fer la síntesi de les regles d'esdeveniments que es referiran a la classe d'esdeveniments estructurals *i_plantillaExcedida*:

i_plantillaExcedida ← aux(plantilla).

Els conjunts de classes d'esdeveniments causa i prerequisit que s'obtenen per fer certa l'única conjunció d'aquesta regla són els següents:

| | causa | prerequisit |
|---|---|---|
| i_plantillaExcedida | {i_plantilla} | {d_plantilla} |
| | {d_plantilla} | {i_plantilla} ♦ |

**Eliminar solucions subsumides.** Les solucions a què s'arriba seguint aquests passos, sobretot en el cas de partir de regles de transició i d'esdeveniments interns sense simplificar, inclouen, en alguns casos, solucions que cal que siguin eliminades.

Es tracta de solucions amb un conjunt de classes d'esdeveniments causa subsumit per conjunts de classes d'esdeveniments causa d'altres solucions. Així, les solucions subsumides han de ser eliminades.

Alhora, els conjunts de classes d'esdeveniments prerequisit corresponents a cada conjunt de classes d'esdeveniments causa que subsumeix les solucions eliminades, s'augmentaran amb les classes d'esdeveniments als conjunts de classes d'esdeveniments causa i dels conjunts de classes d'esdeveniments prerequisit de les solucions eliminades que no estan en aquest conjunt ni en el seu conjunt de classes prerequisit. Això és per assegurar que es disposi de suficient informació per calcular quins esdeveniments han ocorregut de la classe d'esdeveniments estructurals en qüestió.

**Exemple 5.3.19**

Les regles d'esdeveniments que s'obtenen per a les classes d'esdeveniments estructurals *i_plantilla* i *d_plantilla*, un cop eliminades les solucions subsumides, són les següents:

Regles(i_plantilla) = {ri1,ri2}
causa(ri1) = {i_treballaA}.
prerequisit(ri1) = {i_departament}.
causa(ri2) = {i_departament}.
prerequisit(ri2) = {i_treballaA,d_treballaA}.♦

Regles(d_plantilla) = {rd1}
causa(rd1) = {d_treballaA}.
prerequisit(rd1) = {i_treballaA i_departament}.

**Exemple 5.3.20**

Si en comptes d'aplicar el procediment de síntesi a partir de les regles de transició i d'esdeveniments interns de la classe d'esdeveniments estructurals *i_plantilla* sense simplificar, ho fem amb les simplificades, figura 5.3.3, s'obtenen les solucions següents:

| | causa | prerequisit |
|---|---|---|
| i_plantilla | {i_treballaA} | { } |

Per a la classe d'esdeveniments estructurals *d_plantilla*, el resultat és:

| | causa | prerequisit |
|---|---|---|
| d_plantilla | {d_treballaA} | { } |

La interpretació que en podem fer és que en simplificar les regles de transició i d'esdeveniments interns de l'MCDO, s'ha pogut veure que els esdeveniments de la classe *i_departament* no són rellevants ni en la inserció d'un treballador a la plantilla d'un departament ni en la seva eliminació. Les regles d'esdeveniments resultants són:

Regles(i_plantilla) = {ri1,ri2}
causa(ri1) = {i_treballaA}.
prerequisit(ri1) = { }.

Regles(d_plantilla) = {rd1}
causa(rd1) = {d_treballaA}.
prerequisit(rd1) = { }. ♦

**Precedència del càlcul d'atributs derivats d'esdeveniments externs.** Abans de donar les regles d'esdeveniments com a definitives, fa falta un pas més. Concretament, cal tenir en compte la necessitat de mantenir un ordre entre el càlcul dels atributs derivats d'un esdeveniment extern d'una certa classe abans que aquest esdeveniment afecti el sistema. Per tal que aquest ordre es mantingui, caldrà modificar

les regles d'esdeveniments obtingudes i, així, assegurar que els Esquemes d'Interacció resultants el seguiran.

Concretament, en cada regla d'esdeveniments amb una classe d'esdeveniments causa que sigui una classe d'esdeveniments externs, amb un o més atributs derivats, hi haurem d'afegir com a classes d'esdeveniments prerequisit les classes d'esdeveniments estructurals corresponents a cadascun dels atributs derivats. Així assegurarem que els esdeveniments externs de la classe no afectaran cap objecte abans que s'hagi calculat els valors dels seus atributs derivats.

**Exemple 5.3.21**

Al sistema exemple, no hi ha cap classe d'esdeveniments externs que tingui definit un atribut derivat. Ara bé, suposem el cas que a la classe d'esdeveniments externs *nouSou* hi hagués definit l'atribut derivat *souNet* que es calculés a partir de l'atribut *import*. En aquest cas, hauríem d'afegir a cadascuna de les regles d'esdeveniments que tinguessin com una de les seves classes d'esdeveniments causa *nouSou*, la classe d'esdeveniments prerequisit *i_souNet*. ♦

## 5.4 Determinació d'Esquemes d'Interacció per al cas del model exemple

En aquesta secció veurem els Esquemes d'Interacció que s'obtindrien mitjançant el procediment de determinació d'un Esquema d'Interacció vàlid, presentat al capítol 3, per a tres tipus de transaccions corresponents al model exemple de la secció 5.2. Cal recordar que aquest procediment només determina un únic Esquema vàlid, dels molts possibles. Concretament determina l'Esquema d'Interacció vàlid on els tipus d'interaccions estan distribuïdes en un nombre més petit possible d'estats.

Començarem veient el cas de les transaccions del tipus TTR = {nouDirector}. L'Esquema d'Interacció que obtenim amb el procediment és el que trobem gràficament representat a la figura 5.4.1, i que està format pels tipus d'interaccions següents:

EI = {tipusInteracció(nouDirector,nouDirector,1),
tipusInteracció(nouDirector,departament,1),
tipusInteracció(nouDirector,treballador,1),
tipusInteracció(i_responsable,treballador,2),
tipusInteracció(d_responsable,treballador,2),
tipusInteracció(i_treballadorÉsDirector,director,2),
tipusInteracció(d_treballadorÉsDirector,director,2),
tipusInteracció(i_director,director,3),

tipusInteracció(d_director,director,3),
tipusInteracció(i_treballadorCap,director,3),
tipusInteracció(d_treballadorCap,director,3)}

D'aquest Esquema d'Interacció podem destacar el tipus d'interacció tipusInteracció(i_responsable,treballador,2), aquest tipus d'interacció assenyala que, quan es s'assigna un nou responsable d'un departament, això pot afectar l'estat d'un o més objectes de la classe *treballador*.

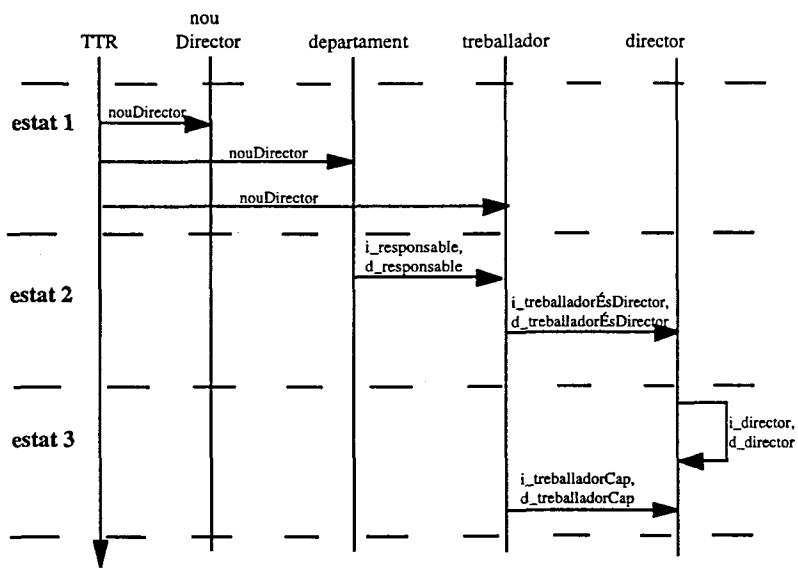


**Figura 5.4.1** Esquema d'Interacció TTR = {nouDirector}

Ara veurem el cas de les transaccions del tipus TTR = {nouSou,nouDirector} L'Esquema d'Interacció que obtenim amb el procediment és el que trobem gràficament representat a la figura 5.4.2, i que està format pels tipus d'interaccions següents:

EI = {tipusInteracció(nouSou,nouSou,1),
tipusInteracció(nouDirector,nouDirector,1),
tipusInteracció(nouDirector,departament,1),
tipusInteracció(nouDirector,treballador,1),
tipusInteracció(i_responsable,treballador,2),
tipusInteracció(d_responsable,treballador,2),
tipusInteracció(i_treballadorÉsDirector,director,2),
tipusInteracció(d_treballadorÉsDirector,director,2),

tipusInteracció(nouSou,treballador,3),
tipusInteracció(d_director,director,3),
tipusInteracció(i_treballadorCap,director,3),
tipusInteracció(d_treballadorCap,director,3),
tipusInteracció(i_treballadorSou,director,4),
tipusInteracció(d_treballadorSou,director,4),
tipusInteracció(i_director,director,4)}

En aquest Esquema d'Interacció es pot veure que els esdeveniments de les classes *i_treballadorSou* i *d_treballadorSou* no s'envien als objectes de la classe *director* fins que s'han aplicat els canvis a l'atribut *sou* provocats pel tipus de transacció.

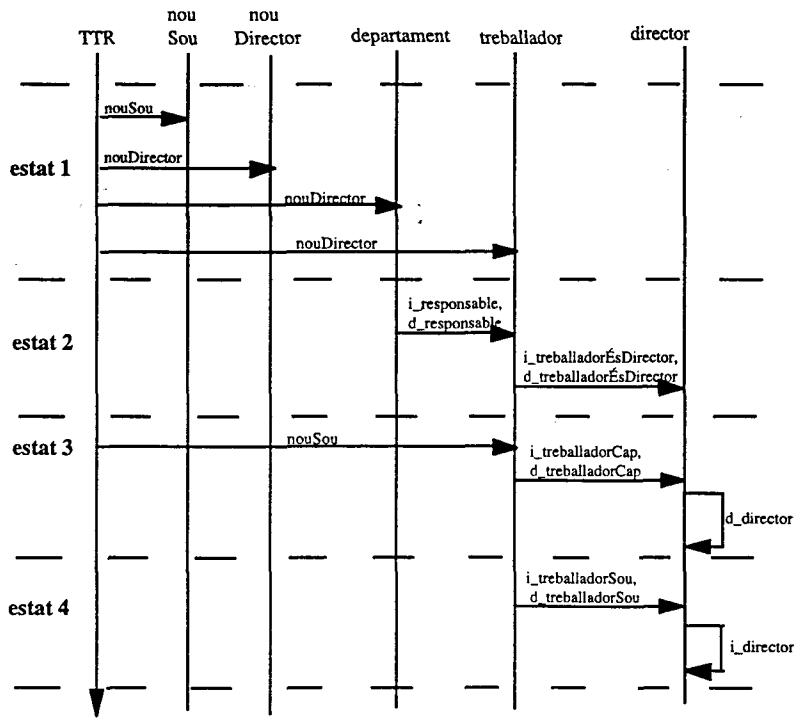


**Figura 5.4.2** Esquema d'Interacció TTR = {nouSou,nouDirector}

Suposem per acabar, el tipus de transacció TTR = {nouTreballador,nouDirector}. L'Esquema d'Interacció que obtenim amb el procediment és el que trobem gràficament representat a la figura 5.4.3, i que està format pels tipus d'interaccions següents:

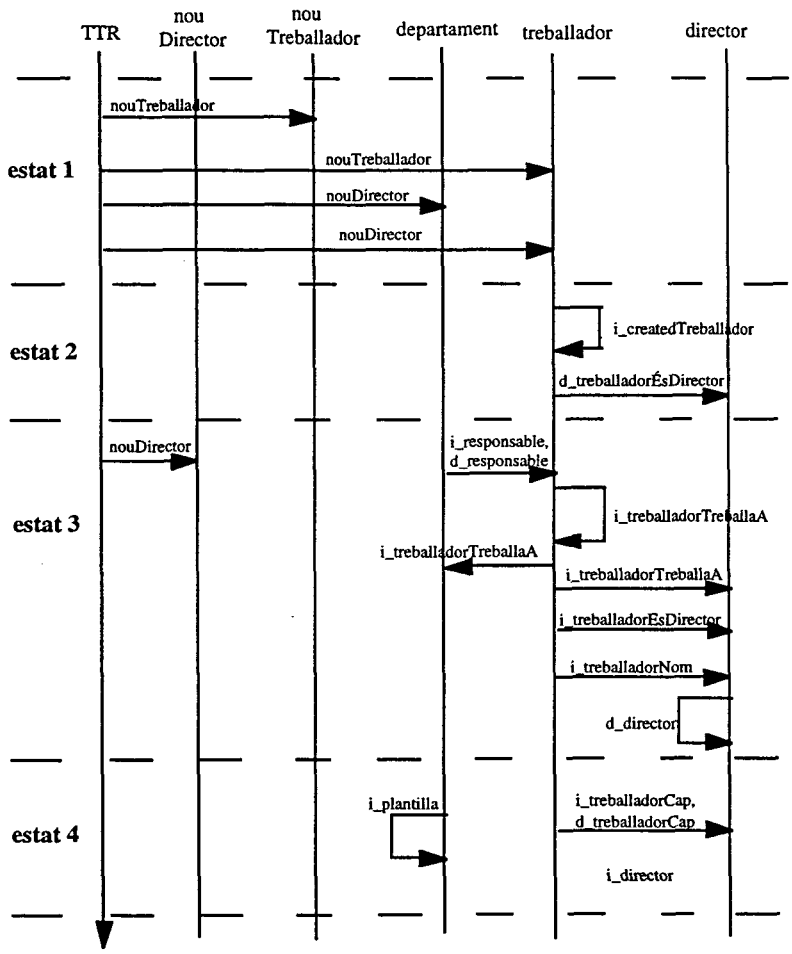


**Figura 5.4.3** Esquema d'Interacció TTR = {nouTreballador,nouDirector}

EI = {tipusInteracció(nouTreballador,nouTreballador,1),
tipusInteracció(nouTreballador,treballador,1),
tipusInteracció(nouDirector,departament,1),
tipusInteracció(nouDirector,treballador,1),
tipusInteracció(i_createdTreballador,treballador,2),
tipusInteracció(d_treballadorÉsDirector,director,2),
tipusInteracció(d_director,director,3)
tipusInteracció(i_responsable,treballador,3),
tipusInteracció(d_responsable,treballador,3),
tipusInteracció(i_treballadorÉsDirector,director,3),

tipusInteracció(i_treballadorNom,director,3),
tipusInteracció(i_treballadorTreballaA,treballador,3),
tipusInteracció(i_treballadorTreballaA,departament,3),
tipusInteracció(i_treballadorTreballaA,director,3),
tipusInteracció(nouDirector,nouDirector,3),
tipusInteracció(i_treballadorCap,director,4),
tipusInteracció(d_treballadorCap,director,4),
tipusInteracció(i_plantilla,departament,4),
tipusInteracció(i_director,director,4)}

En aquest Esquema d'Interacció es pot destacar que quan s'insereix un nou treballador a la plantilla d'un departament això pot afectar altres parts de l'estat d'un o més objectes de la classe *departament.*

# Annex 1

Aquesta secció mostra el resultat complet de la síntesi feta a partir de l'MCDO del sistema exemple (secció 5.2). Tenint en compte que per fer la síntesi de regles d'esdeveniments són necessàries les regles de transició i d'esdeveniments interns, abans de donar la síntesi de regles d'esdeveniments s'inclouen les regles de transició i d'esdeveniments interns simplificades que es poden generar a partir de l'MCDO (vegeu més endavant).

**Classes d'esdeveniments externs**
EsdExterns = {nouTreballador, baixaTreballador, nouDepartament,
canviDepartament, nouSou, nouDirector}

**Classes d'objectes**
Objectes = {treballador, departament, director,
nouTreballador, baixaTreballador, nouDepartament,
canviDepartament, nouSou, nouDirector}

**Atributs d'una classe d'objectes**
Atributs(treballador) = {treballadorNom, treballadorÉsDirector,
treballadorTreballaA, treballadorCap, treballadorSou}
Atributs(departament) = {responsable, plantilla}
Atributs(director) = {sobreSou, directorNom, directorÉsDirector,
directorTreballaA, directorCap, directorSou}
Atributs(nouTreballador) = { }
Atributs(baixaTreballador) = { }
Atributs(nouDepartament) = { }
Atributs(canviDepartament) = { }
Atributs(nouSou) = { }
Atributs(nouDirector) = { }

**Classes d'esdeveniments generats associades a una classe d'objectes**
Generacions(treballador) = {important,createdTreballador,destroyedTreballador}
Generacions(departament) = {createdDepartament,destroyedDepartament}
Generacions(director) = { }
Generacions(nouTreballador) = { }
Generacions(baixaTreballador) = { }
Generacions(nouDepartament) = { }
Generacions(canviDepartament) = { }
Generacions(nouSou) = { }
Generacions(nouDirector) = { }

**Restriccions associades a una classe d'objectes**
Restriccions(treballador) = { }
Restriccions(departament) = {plantillaExcedida}

Restriccions(director) = { }
Restriccions(nouTreballador) = {ic1}
ic1(D,T) ← nouTreballador(Z,T),dept(Z:D),departament(D,T).
Restriccions(baixaTreballador) = {ic2}
ic2(E,T) ← baixaTreballador(Z,T),treb(Z:E),not(treballador(E,T-1)).
Restriccions(nouDepartament) = { }
Restriccions(canviDepartament) = {ésElDirector,ic3,ic4}
ic3(E,T) ← canviDepartament(Z,T),treb(Z:E),not(treballador(E,T)).
ic4(D,T) ← canviDepartament(Z,T),dept(Z:D),not(departament(D,T)).
Restriccions(nouSou) = {ic5}
ic5(E,T) ← nouSou(Z,T),treb(Z:E),not(treballador(E,T)).
Restriccions(nouDirector) = {jaÉsElDirector,noTreballaA,ic6,ic7}
ic6(E,T) ← nouDirector(Z,T),dir(Z:E),not(treballador(E,T)).
ic7(D,T) ← nouDirector(Z,T),dept(Z:D),not(departament(D,T)).

**Classes d'esdeveniments estructurals d'inserció d'objectes**
Inserció(treballador) = i_treballador
Inserció(departament) = i_departament
Inserció(director) = i_director
Inserció(nouTreballador) = -
Inserció(baixaTreballador) = -
Inserció(nouDepartament) = -
Inserció(canviDepartament) = -
Inserció(nouSou) = -
Inserció(nouDirector) = -

**Classes d'esdeveniments estructurals d'esborrat d'objectes**
Esborrat(treballador) = d_treballador
Esborrat(departament) = d_departament
Esborrat(director) = d_director
Esborrat(nouTreballador) = -
Esborrat(baixaTreballador) = -
Esborrat(nouDepartament) = -
Esborrat(canviDepartament) = -
Esborrat(nouSou) = -
Esborrat(nouDirector) = -

**Classes d'esdeveniments estructurals de modificació dels valors d'atributs**
Modificació(treballadorNom) = {i_treballadorNom,d_treballadorNom}
Modificació(treballadorÉsDirector) = {i_treballadorÉsDirector,
d_treballadorÉsDirector}
Modificació(treballadorTreballaA) = {i_treballadorTreballaA,
d_treballadorTreballaA}
Modificació(treballadorCap) = {i_treballadorCap,d_treballadorCap}
Modificació(treballadorSou) = {i_treballadorSou,d_treballadorSou}
Modificació(responsable) = {i_responsable,d_responsable}
Modificació(plantilla) = {i_plantilla,d_plantilla}
Modificació(sobreSou) = {i_sobreSou,d_sobreSou}
Modificació(directorNom) = {i_directorNom,d_directorNom}
Modificació(directorÉsDirector) = {i_directorÉsDirector,
d_directorÉsDirector}
Modificació(directorTreballaA) = {i_directorTreballaA,
d_directorTreballaA}
Modificació(directorCap) = {i_directorCap,d_directorCap}
Modificació(directorSou) = {i_directorSou,d_directorSou}

**Classes d'esdeveniments estructurals de violació de restriccions d'integritat**

Restricció(plantillaExcedida) = i_plantillaExcedida
Restricció(ic1) = i_ic1
Restricció(ic2) = i_ic2
Restricció(ésElDirector) = i_ésElDirector
Restricció(ic3) = i_ic3
Restricció(ic4) = i_ic4
Restricció(ic5) = i_ic5
Restricció(jaÉsElDirector) = i_jaÉsElDirector
Restricció(noTreballaA) = i_noTreballaA
Restricció(ic6) = i_ic6
Restricció(ic7) = i_ic7

**Classes d'esdeveniments estructurals de generació d'esdeveniments**

Generació(important) = i_important
Generació(createdTreballador) = i_createdTreballador
Generació(destroyedTreballador) = i_destroyedTreballador
Generació(createdDepartament) = i_createdDepartament
Generació(destroyedDepartment) = i_destroyedDepartament

**Regles de transició i d'esdeveniments interns**

A continuació trobem les regles de transició i d'esdeveniments interns. En elles hem subratllat els predicats que són rellevants per a la síntesi de les regles d'esdeveniments.

```
i_ic1(D,T) ← nouTreballador(Z,T),dept(Z:D),
                    not(departament(D,T-1)),not(i_departament(D,T)).

i_ic2(E,T) ← baixaTreballador(Z,T),nom(Z:E),not(treballador(E,T-1)).

i_ésElDirector(E,T) ← canviDepartament(Z,T),treb(Z:E),dept(Z:D),
                    responsable(D,T-1:E),not(d_responsable(D,T:E)).
i_ésElDirector(E,T) ← canviDepartament(Z,T),treb(Z:E),dept(Z:D),i_responsable(D,T:E).
i_ic3(E,T) ← canviDepartament(Z,T),treb(Z:E),not(treballador(E,T-1)),not(i_treballador(E,T)).
i_ic3(E,T) ← canviDepartament(Z,T),treb(Z:E),d_treballador(E,T).
i_ic4(D,T) ← canviDepartament(Z,T),dept(Z:D),
                    not(departament(D,T-1)),not(i_departament(D,T)).

i_ic5(E,T) ← nouSou(Z,T),treb(Z:E),not(treballador(E,T-1)),not(i_treballador(E,T)).
i_ic5(E,T) ← nouSou(Z,T),treb(Z:E),d_treballador(E,T).

i_jaÉsElDirector(E,T) ← nouDirector(Z,T),dir(Z:E),dept(Z:D),responsable(D,T-1:E).
i_noTreballaA(D,E,T) ← nouDirector(Z,T),dir(Z:E),dept(Z:D),
                    not(treballadorTreballaA(E,T-1:D)),not(i_treballadorTreballaA(E,T:D)).
i_noTreballaA(D,E,T) ← nouDirector(Z,T),dir(Z:E),dept(Z:D),d_treballadorTreballaA(E,T:D).
i_ic6(E,T) ← nouDirector(Z,T),dir(Z:E),not(treballador(E,T-1)),not(i_treballador(E,T)).
i_ic6(E,T) ← nouDirector(Z,T),dir(Z:E),d_treballador(E,T).
i_ic7(D,T) ← nouDirector(Z,T),dept(Z:D),not(departament(D,T-1)),not(i_departament(D,T)).
```

Classes d'esdeveniments externs

```
i_departament(D,T) ← i_createdDepartament(oid:D,time:T).

i_createdDepartament(oid:D,time:T) ← nouDepartament(Z,T).

i_responsable(D,T:M)  ← nouDirector(Z,T),dept(Z:D),dir(Z:M).
d_responsable(D,T:M)  ← nouDirector(Z,T),dept(Z:D),responsable(D,T-1:M).

i_plantilla(D,T:E) ← i_treballadorTreballaA(E,T:D).
d_plantilla(D,T:E) ← d_treballadorTreballaA(E,T:D), treballadorTreballaA(E,T-1:D).

i_plantillaExcedida(D,N,T) ← count(E,plantilla(D,T-1:E),NA),
          count(E,i_plantilla(D,T:E),NAFEG),count(E,d_plantilla(D,T:E),NELIM),
            N = NA + NAFEG - NELIM, N > 50.
```

Classe d'objectes *departament*

```
i_treballador(E,T) ← i_createdTreballador(oid:E, nom:N, deptInicial: D,time:T).
d_treballador(E,T) ← i_destroyedTreballador(oid:E, time:T).

i_createdTreballador(oid:E, nom:N, deptInicial: D, time:T) ←
                                  nouTreballador(Z,T), nom(Z:N), dept(Z:D).
i_destroyedTreballador(oid:E,time:T) ← baixaTreballador(Z,T), treb(Z:E).

i_treballadorNom(E,T:N) ← i_createdTreballador(E,T),nom(E:N).
d_treballadorNom(E,T:N) ← d_treballador(E,T).

i_treballadorÉsDirector(E,T:D) ← nouDirector(Z,T),dept(Z:D),dir(Z:E).
d_treballadorÉsDirector(E,T:D) ← nouDirector(Z,T),dept(Z:D),treballadorÉsDirector(E,T-1:D).
d_treballadorÉsDirector(E,T:D) ← d_treballador(E,T).

i_treballadorTreballaA(E,T:D) ← i_createdTreballador(E,T),deptInicial(E:D).
i_treballadorTreballaA(E,T:D) ← canviDepartament(Z,T),treb(Z:E),dept(Z:D).
d_treballadorTreballaA(E,T:D) ← canviDepartament(Z,T),treb(Z:E),dept(Z:D2),
                                  D2≠D,treballadorTreballaA(E,T-1:D).
d_treballadorTreballaA(E,T:D) ← d_treballador(E,T).

i_treballadorCap(E,T:M) ← treballadorCap(E,T:M), not(treballadorCap(E,T-1:M)).
d_treballadorCap(E,T:M) ← treballadorCap(E,T-1:M),not(treballadorCap(E,T:M)).

i_treballadorSou(E,T:S) ← nouSou(Z,T),treb(Z:E),import(Z:S).
d_treballadorSou(E,T:S) ← nouSou(Z,T),treb(Z:E),treballadorSou(E,T-1:S).
d_treballadorSou(E,T:S) ← d_treballador(E,T).

i_important(treb:E,total:S,temps:T) ← nouSou(Z,T),treb(Z:E),import(Z:S),
                                      d_director(E,T),S>50000.
i_important(treb:E,total:S,temps:T) ← nouSou(Z,T),treb(Z:E),import(Z:S),
                                    not(director(E,T-1)),not(i_director(E,T)),S>50000.

treballadorCap(E,T:M) ← treballadorTreballaA(E,T-1:D),not(d_treballadorTreballaA(E,T:D)),
                        responsable(E,T-1:M),not(d_responsable(E,T:M)).
treballadorCap(E,T:M) ← i_treballadorTreballaA(E,T:D),
                        responsable(E,T-1:M),not(d_responsable(E,T:M)).
treballadorCap(E,T:M) ← treballadorTreballaA(E,T-1:D),not(d_treballadorTreballaA(E,T:D)),
                          i_responsable(D,T:M).
treballadorCap(E,T:M) ← i_treballadorTreballaA(E,T:D),i_responsable(D,T:M).
```

Classe d'objectes *treballador*

```
i_director(M,T) ← i_treballadorÉsDirector(M,T).
i_director(M,T) ← i_treballador(M,T),i_treballadorÉsDirector(M,T).
i_director(M,T) ← d_treballadorÉsDirector(M,T).

i_sobreSou(M,T:SS) ← i_treballadorSou(M,T:S), SS = S*20/100,
                     not(sobreSou(M,T-1:SS)),director(M,T-1),not(d_director(M,T)).
i_sobreSou(M,T:SS) ← i_director(M,T), treballadorSou(M,T-1:S),
                     SS = S*20/100, not(d_treballadorSou(M,T:S)).
i_sobreSou(M,T:SS) ← i_treballadorSou(M,T:S), SS = S*20/100, i_director(M,T).
d_sobreSou(M,T:SS) ← d_treballadorSou(M,T:S), sobreSou(M,T-1:SS).
d_sobreSou(M,T:SS) ← d_director(M,T),sobreSou(M,T-1:SS).

i_directorNom(M,T:N) ← i_treballadorNom(M,T:N).
d_directorNom(M,T:N) ← d_treballadorNom(M,T:N).

i_directorÉsDirector(M,T:D) ← i_treballadorÉsDirector(M,T:D).
d_directorÉsDirector(M,T:D) ← d_treballadorÉsDirector(M,T:D).

i_directorTreballaA(M,T:D) ← i_treballadorTreballaA(M,T:D).
d_directorTreballaA(M,T:D) ← d_treballadorTreballaA(M,T:D).

i_directorCap(M,T:D) ← i_treballadorCap(M,T:D).
d_directorCap(M,T:D) ← d_treballadorCap(M,T:D).

i_directorSou(M,T:D) ← i_treballadorSou(M,T:D).
d_directorSou(M,T:D) ← d_treballadorSou(M,T:D).
```

Classe d'objectes *director*

**Regles d'esdeveniments que es refereixen a classes d'esdeveniments estructurals d'inserció d'objectes**

Regles(i_treballador) = $\{r_1\}$
causa($r_1$) = {i_createdTreballador}
prerequisit($r_1$) = { }
Regles(i_departament) = $\{r_2\}$
causa($r_2$) = {i_createdDepartament}
prerequisit($r_2$) = { }
Regles(i_director) = $\{r_3\}$
causa($r_3$) = {i_treballadorÉsDirector}
prerequisit($r_3$) = {i_treballador}

**Regles d'esdeveniments que es refereixen a classes d'esdeveniments estructurals d'esborrat d'objectes**

Regles(d_treballador) = $\{r_4\}$
causa($r_4$) = {i_destroyedTreballador}
prerequisit($r_4$) = { }
Regles(d_departament) = { }
Regles(d_director) = $\{r_5\}$
causa($r_5$) = {d_treballadorÉsDirector}
prerequisit($r_5$) = { }

**Regles d'esdeveniments que es refereixen a classes d'esdeveniments estructurals de modificació dels valors d'atributs**

Regles(i_treballadorNom) = {$r_6$}
causa($r_6$) = {i_createdTreballador}
prerequisit($r_6$) = { }
Regles(d_treballadorNom) = {$r_7$}
causa($r_7$) = {d_treballador}
prerequisit($r_7$) = { }
Regles(i_treballadorÉsDirector) = {$r_8$}
causa($r_8$) = {nouDirector}
prerequisit($r_8$) = {i_jaÉsElDirector,i_noTreballaA,i_ic6,i_ic7}
Regles(d_treballadorÉsDirector) = {$r_9$,$r_{10}$}
causa($r_9$) = {nouDirector}
prerequisit($r_9$) = {i_jaÉsElDirector,i_noTreballaA,i_ic6,i_ic7}
causa($r_{10}$) = {d_treballador}
prerequisit($r_{10}$) = { }
Regles(i_treballadorTreballaA) = {$r_{11}$,$r_{12}$}
causa($r_{11}$) = {i_createdTreballador}
prerequisit($r_{11}$) = { }
causa($r_{12}$) = {canviDepartament}
prerequisit($r_{12}$) = {i_ésElDirector,i_ic3,i_ic4}
Regles(d_treballadorTreballaA) = {$r_{13}$,$r_{14}$}
causa($r_{13}$) = {canviDepartament}
prerequisit($r_{13}$) = {i_ésElDirector,i_ic3,i_ic4}
causa($r_{14}$) = {d_treballador}
prerequisit($r_{14}$) = { }
Regles(i_treballadorCap) = {$r_{15}$,$r_{16}$}
causa($r_{15}$) = {i_treballadorTreballaA}
prerequisit($r_{15}$) = {d_treballadorTreballaA,i_responsable,d_responsable}
causa($r_{16}$) = {i_responsable}
prerequisit($r_{16}$) = {d_responsable,i_treballadorTreballaA,
d_treballadorTreballaA}
Regles(d_treballadorCap) = {$r_{17}$,$r_{18}$}
causa($r_{17}$) = {d_treballadorTreballaA}
prerequisit($r_{17}$) = {i_treballadorTreballaA,i_responsable,d_responsable}
causa($r_{18}$) = {d_responsable}
prerequisit($r_{18}$) = {i_responsable,i_treballadorTreballaA,
d_treballadorTreballaA}
Regles(i_treballadorSou) = {$r_{19}$}
causa($r_{19}$) = {nouSou}
prerequisit($r_{19}$) = {i_ic5}
Regles(d_treballadorSou) = {$r_{20}$,$r_{21}$}
causa($r_{20}$) = {nouSou}
prerequisit($r_{20}$) = {i_ic5}
causa($r_{21}$) = {d_treballador}
prerequisit($r_{21}$) = { }
Regles(i_responsable) = {$r_{22}$}
causa($r_{22}$) = {nouDirector}
prerequisit($r_{22}$) = {i_jaÉsElDirector,i_noTreballaA,i_ic6,i_ic7}
Regles(d_responsable) = {$r_{23}$}
causa($r_{23}$) = {nouDirector}
prerequisit($r_{23}$) = {i_jaÉsElDirector,i_noTreballaA,i_ic6,i_ic7}
Regles(i_plantilla) = {$r_{24}$}
causa($r_{24}$) = {i_treballadorTreballaA}
prerequisit($r_{24}$) = { }
Regles(d_plantilla) = {$r_{25}$}
causa($r_{25}$) = {d_treballadorTreballaA}

prerequisit($r_{25}$) = { }
Regles(i_sobreSou) = {$r_{26}$,$r_{27}$}
causa($r_{26}$) = {i_treballadorSou}
prerequisit($r_{26}$) = {i_director,d_director}
causa($r_{27}$) = {i_director}
prerequisit($r_{27}$) = {i_treballadorSou,d_treballadorSou}
Regles(d_sobreSou) = {$r_{28}$,$r_{29}$}
causa($r_{28}$) = {d_treballadorSou}
prerequisit($r_{28}$) = { }
causa($r_{29}$) = {d_director}
prerequisit($r_{29}$) = { }
Regles(i_directorNom) = {$r_{30}$}
causa($r_{30}$) = {i_treballadorNom}
prerequisit($r_{30}$) = { }
Regles(d_directorNom) = {$r_{31}$}
causa($r_{31}$) = {d_treballadorNom}
prerequisit($r_{31}$) = { }
Regles(i_directorÉsDirector) = {$r_{32}$}
causa($r_{32}$) = {i_treballadorÉsDirector}
prerequisit($r_{32}$) = { }
Regles(d_directorÉsDirector) = {$r_{33}$}
causa($r_{33}$) = {d_treballadorÉsDirector}
prerequisit($r_{33}$) = { }
Regles(i_directorTreballaA) = {$r_{34}$}
causa($r_{34}$) = {i_treballadorTreballaA}
prerequisit($r_{34}$) = { }
Regles(d_directorTreballaA) = {$r_{35}$}
causa($r_{35}$) = {d_treballadorTreballaA}
prerequisit($r_{35}$) = { }
Regles(i_directorCap) = {$r_{36}$}
causa($r_{36}$) = {i_treballadorCap}
prerequisit($r_{36}$) = { }
Regles(d_directorCap) = {$r_{37}$}
causa($r_{37}$) = {d_treballadorCap}
prerequisit($r_{37}$) = { }
Regles(i_directorSou) = {$r_{38}$}
causa($r_{38}$) = {i_treballadorSou}
prerequisit($r_{38}$) = { }
Regles(d_directorSou) = {$r_{39}$}
causa($r_{39}$) = {d_treballadorSou}
prerequisit($r_{39}$) = { }

**Regles d'esdeveniments que es refereixen a classes d'esdeveniments estructurals de generació d'esdeveniments**

Regles(i_important) = {$r_{40}$}
causa($r_{40}$) = {nouSou}
prerequisit($r_{40}$) = {i_director,d_director,i_ic5}
Regles(i_createdTreballador) = {$r_{41}$}
causa($r_{41}$) = {nouTreballador}
prerequisit($r_{41}$) = {i_ic1}
Regles(i_destroyedTreballador) = {$r_{42}$}
causa($r_{42}$) = {baixaTreballador}
prerequisit($r_{42}$) = {i_ic2}
Regles(i_createdDepartament) = {$r_{43}$}
causa($r_{43}$) = {nouDepartament}
prerequisit($r_{43}$) = { }
Regles(i_destroyedDepartament) = { }

**Regles d'esdeveniments que es refereixen a classes d'esdeveniments estructurals de violació de restriccions d'integritat**

Regles(i_plantillaExcedida) = {$r_{44}$,$r_{45}$}
causa($r_{44}$) = {i_plantilla}
prerequisit($r_{44}$) = {d_plantilla}
causa($r_{45}$) = {d_plantilla}
prerequisit($r_{45}$) = {i_plantilla}

Regles(i_ic1) = {$r_{46}$}
causa($r_{46}$) = {nouTreballador}
prerequisit($r_{46}$) = {i_departament}

Regles(i_ic2) = {$r_{47}$}
causa($r_{47}$) = {baixaTreballador}
prerequisit($r_{47}$) = {}

Regles(i_ésElDirector) = {$r_{48}$}
causa($r_{48}$) = {canviDepartament}
prerequisit($r_{48}$) = {i_responsable,d_responsable}

Regles(i_ic3) = {$r_{49}$}
causa($r_{49}$) = {canviDepartament}
prerequisit($r_{49}$) = {i_treballador,d_treballador}

Regles(i_ic4) = {$r_{50}$}
causa($r_{50}$) = {canviDepartament}
prerequisit($r_{50}$) = {i_departament}

Regles(i_ic5) = {$r_{51}$}
causa($r_{51}$) = {nouSou}
prerequisit($r_{51}$) = {i_treballador,d_treballador}

Regles(i_jaÉsElDirector) = {$r_{52}$}
causa($r_{52}$) = {nouDirector}
prerequisit($r_{52}$) = {}

Regles(i_noTreballaA) = {$r_{53}$}
causa($r_{53}$) = {nouDirector}
prerequisit($r_{53}$) = {i_treballadorTreballaA,d_treballadorTreballaA}

Regles(i_ic6) = {$r_{54}$}
causa($r_{54}$) = {nouDirector}
prerequisit($r_{54}$) = {i_treballador,d_treballador}

Regles(i_ic7) = {$r_{55}$}
causa($r_{55}$) = {nouDirector}
prerequisit($r_{55}$) = {i_departament}

# Annex 2

**Definició de dimensió positiva d'un predicat [San94]**

*"Dimensió positiva. Sigui p un predicat i k un vector de constants. Suposem que el fet p(k) és cert a l'instant T-1. Què podem dir sobre el valor de veritat de p(k) a l'instant T? Hi ha tres casos possibles:*

*(a) p(k) serà cert a T. Llavors diem que p és P-permanent.*

*(b) p(k) serà fals a T. Llavors diem que p és P-momentani.*

*(c) p(k) pot ser cert o fals a T. En aquest darrer cas, assumim que no ha succeït cap esdeveniment extern a l'instant T. Llavors trobem tres subcasos:*

*(c1) p(k) serà cert a T. Llavors diem que p és P-estat.*

*(c2) p(k) serà fals a T. Llavors diem que p és P-transitori.*

*(c3) p(k) pot ser cert o fals a T, segons el valor de veritat d'un predicat $\phi(k,T)$. Llavors diem que p és P-espontani."*

**Caracterització de la dimensió positiva dels predicats en un MCDO segons el paper que fan en el model**

En els MCDO hi ha predicats que només tenint en compte el paper que fan en el model es poden classificar en la seva dimensió positiva. Concretament:

- Els predicats que fan referència a l'ocurrència d'esdeveniments externs, esdeveniments de naixement, esdeveniments de mort i esdeveniments generats, són sempre predicats P-momentani. Això és així perquè si ha ocorregut un d'aquests esdeveniments en un instant T-1, podem afirmar que aquest esdeveniment no tornarà a ocórrer en l'instant T.

- Els predicats que fan referència a l'existència d'un objecte en una classe extensional, a l'existència d'un objecte en una classe intensional, a l'existència d'un objecte en una classe generalització, són predicats P-estat o P-permanent. Són P-estat en el cas que l'objecte no pugui deixar d'existir a la classe i P-permanent en el cas contrari.

- Els predicats que fan referència al valor d'un atribut d'un objecte són P-permanent en el cas que l'atribut sigui un atribut de valor constant definit en una classe en què els objectes no moren mai, i són P-estat en la resta de casos.

- Finalment, els predicats de valor d'un atribut d'un esdeveniment, els predicats avaluables i els predicats d'inconsistència no té sentit classificar-los. Els primers i els segons perquè no tenen entre els seus atributs un de domini Time que faci referència a l'instant de temps en què l'esdeveniment ha ocorregut. Els tercers, tal com es diu a [Oli89], perquè l'estat del sistema està sempre en un estat consistent i els esdeveniments que puguin violar aquesta consistència seran rebutjats.

Només resten uns predicats que han de ser classificats pel dissenyador, estudiant-ne la semàntica particular, que són els predicats auxiliars que apareixen en l'MCDO com a resultat de passar les regles a forma normal.

## Transformació de literals ordinaris

| | | P-tipus dels predicats | P(Lr) | Pr(Lr) | N(Lr) |
|---|---|---|---|---|---|
| Lr-POSITIU | Lr-ACTUAL | P-estat | fals | obj(O,T-1) ∧ ¬ **idestroyed** obj(O,T)<br>subobj(O,T-1) ∧ ¬ δsubobj(O,T)<br>superobj(O,T-1) ∧ ¬ δsuperobj(O,T)<br>atr_obj(O,T-1:V) ∧ ¬ δatr_obj(O,T:V) | **icreated** obj(O,T)<br>isubobj(O,T)<br>isuperobj(O,T)<br>iatr_obj(O,T:V) |
| | | P-permanent | fals | obj(O,T-1)<br>subobj(O,T-1)<br>superobj(O,T-1)<br>atr_obj(O,T-1:V) | **icreated** obj(O,T)<br>isubobj(O,T)<br>isuperobj(O,T)<br>iatr_obj(O,T:V) |
| | | P-momentani o P-transitori | fals | fals | **icreated** obj(O,T)<br>**idestroyed** obj(O,T)<br>iincons_obj(X,T)<br>igen_obj(X,T)<br>iesdev(Z,T)<br>iincons_esdev(X,T) |
| | Lr-ANTIC | P-estat | T1<T ∧ obj(O,T1)<br>T1<T ∧ subobj(O,T1)<br>T1<T ∧ superobj(O,T1)<br>T1<T ∧ atr_obj(O,T1:V) | T1=T ∧ obj(O,T-1) ∧ ¬ **idestroyed** obj(O,T)<br>T1=T ∧ subobj(O,T-1) ∧ ¬ δsubobj(O,T)<br>T1=T ∧ superobj(O,T-1) ∧ ¬ δsuperobj(O,T)<br>T1=T ∧ atr_obj(O,T-1:V) ∧ ¬ δatr_obj(O,T:V) | T1=T ∧ **icreated** obj(O,T)<br>T1=T ∧ isubobj(O,T)<br>T1=T ∧ isuperobj(O,T)<br>T1=T ∧ iatr_obj(O,T:V) |
| | | P-permanent | T1<T ∧ obj(O,T1)<br>T1<T ∧ subobj(O,T1)<br>T1<T ∧ superobj(O,T1)<br>T1<T ∧ atr_obj(O,T1:V) | T1=T ∧ obj(O,T-1)<br>T1=T ∧ subobj(O,T-1)<br>T1=T ∧ superobj(O,T-1)<br>T1=T ∧ atr_obj(O,T-1:V) | T1=T ∧ **icreated** obj(O,T)<br>T1=T ∧ isubobj(O,T)<br>T1=T ∧ isuperobj(O,T)<br>T1=T ∧ iatr_obj(O,T:V) |
| | | P-momentani o P-transitori | T1<T ∧ **created** obj(O,T1)<br>T1<T ∧ **destroyed** obj(O,T1)<br>T1<T ∧ incons_obj(X,T1)<br>T1<T ∧ gen_obj(X,T1)<br>T1<T ∧ esdev(Z,T1)<br>*T1<T ∧ incons_esdev(X,T1)* | fals | T1=T ∧ **icreated** obj(O,T)<br>T1=T ∧ **idestroyed** obj(O,T)<br>T1=T ∧ iincons_obj(X,T)<br>T1=T ∧ igen_obj(X,T)<br>T1=T ∧ iesdev(Z,T)<br>*T1=T ∧ iincons_esdev(X,T)* |
| Lr-NEGATIU | Lr-ACTUAL | P-estat | fals | ¬ obj(O,T-1) ∧ ¬ **icreated** obj(O,T)<br>¬ subobj(O,T-1) ∧ ¬ isubobj(O,T)<br>¬ superobj(O,T-1) ∧ ¬ isuperobj(O,T)<br>¬ atr_obj(O,T-1:V) ∧ ¬ iatr_obj(O,T:V) | **idestroyed** obj(O,T)<br>δsubobj(O,T)<br>δsuperobj(O,T)<br>δatr_obj(O,T:V) |
| | | P-permanent | fals | ¬ obj(O,T-1) ∧ ¬ **icreated** obj(O,T)<br>¬ subobj(O,T-1) ∧ ¬ isubobj(O,T)<br>¬ superobj(O,T-1) ∧ ¬ isuperobj(O,T)<br>¬ atr_obj(O,T-1:V) ∧ ¬ iatr_obj(O,T:V) | fals |
| | | P-momentani o P-transitori | fals | fals | **icreated** obj(O,T)<br>**idestroyed** obj(O,T)<br>iincons_obj(X,T)<br>igen_obj(X,T)<br>iesdev(Z,T)<br>iincons_esdev(X,T) |
| | Lr-ANTIC | P-estat | T1<T ∧ ¬ obj(O,T1)<br>T1<T ∧ ¬ subobj(O,T1)<br>T1<T ∧ ¬ superobj(O,T1)<br>T1<T ∧ ¬ atr_obj(O,T1:V) | T1=T ∧ ¬ obj(O,T-1) ∧ ¬ **icreated** obj(O,T)<br>T1=T ∧ ¬ subobj(O,T-1) ∧ ¬isubobj(O,T)<br>T1=T ∧ ¬ superobj(O,T-1) ∧ ¬isuperobj(O,T)<br>T1=T ∧ ¬ atr_obj(O,T-1:V)∧ ¬iatr_obj(O,T:V) | T1=T ∧ **idestroyed** obj(O,T)<br>T1=T ∧ δsubobj(O,T)<br>T1=T ∧ δsuperobj(O,T)<br>T1=T ∧ δatr_obj(O,T:V) |
| | | P-permanent | T1<T ∧ ¬ obj(O,T1)<br>T1<T ∧ ¬ subobj(O,T1)<br>T1<T ∧ ¬ superobj(O,T1)<br>T1<T ∧ ¬ atr_obj(O,T1:V) | T1=T ∧ ¬ obj(O,T-1) ∧ ¬ **icreated** obj(O,T)<br>T1=T ∧ ¬ subobj(O,T-1) ∧ ¬ isubobj(O,T)<br>T1=T ∧ ¬ superobj(O,T-1) ∧ ¬ isuperobj(O,T)<br>T1=T ∧ ¬ atr_obj(O,T-1:V) ∧ ¬ iatr_obj(O,T:V) | fals |
| | | P-momentani o P-transitori | T1<T ∧ ¬ **created** obj(O,T1)<br>T1<T ∧ ¬ **destroyed** obj(O,T1)<br>T1<T ∧ ¬ incons_obj(X,T1)<br>T1<T ∧ ¬ gen_obj(X,T1)<br>T1<T ∧ ¬ esdev(Z,T1)<br>T1<T ∧ ¬ incons_esdev(X,T1) | fals | T1=T ∧ ¬ **icreated** obj(O,T)<br>T1=T ∧ ¬ **idestroyed** obj(O,T)<br>T1=T ∧ ¬ incons_obj(X,T)<br>T1=T ∧ ¬ igen_obj(X,T)<br>T1=T ∧ ¬ iesdev(Z,T)<br>T1=T ∧ ¬ incons_esdev(X,T) |

# 6. Us addicional de la síntesi per obtenir documentació complementària sobre un model

La síntesi d'elements d'un model, a part de ser necessària per a la determinació d'Esquemes d'Interacció, pot servir per obtenir una documentació complementària sobre el model que pot ajudar encara més a entendre el seu comportament.

Aquesta documentació complementària es pot obtenir formulant preguntes que seran contestades a partir de la síntesi. Tal com passa en el cas del mètode per determinar Esquemes d'Interacció, les preguntes que es poden formular i les respostes corresponents es mouen en el món del "potencial". És a dir, no es tracta de respondre a preguntes sobre objectes o esdeveniments específics, sinó sobre classes d'objectes i classes d'esdeveniments, i tampoc no es tracta de respondre a preguntes sobre transaccions concretes, sinó sobre tipus de transaccions.

Ja hem fet notar en parlar del mètode per a la determinació d'Esquemes d'Interacció, que aquest fet és alhora un avantatge i un inconvenient. És un inconvenient per la manca de respostes concretes, atès que ens movem en el nivell de classes i tipus. Però alhora és un avantatge, ja que en moure'ns en aquest nivell, aconseguim un alt grau d'abstracció, que no es té en validar un model únicament a partir de casos concrets.

Aquí veurem una mostra de les preguntes que poden ser contestades, classificades segons els elements d'un model a què fan referència. Per a cada una definirem en lògica de primer ordre quina és la resposta, en funció de la síntesi d'elements d'un model, i donarem un exemple de la seva utilització per a la validació del model exemple utilitzat

al capítol 5 d'aquesta tesi. Aquest model s'emmarca en el món de la gestió de personal d'una empresa (la seva descripció i la síntesi dels seus elements estan a la secció 5.2 i a l'annex 2 del capítol 5).

Aquest capítol està dividit en cinc seccions. La secció 6.1 comença amb preguntes que tracten sobre els canvis en l'estat dels objectes. A la secció 6.2 descriurem algunes de les preguntes que es poden fer sobre la modificació del valor dels atributs. A les seccions 6.3 i 6.4 hi ha una mostra de preguntes sobre la generació d'esdeveniments i la violació de restriccions d'integritat. I, per acabar, a la secció 6.5 es plantegen preguntes sobre canvis d'estat concrets.

## 6.1 Preguntes sobre els canvis en l'estat dels objectes

A un dissenyador li pot interessar saber si un tipus de transacció pot canviar l'estat d'un o més objectes d'una certa classe; també li pot interessar saber a quines classes d'objectes pertanyen els objectes als quals pot canviar l'estat una transacció d'un tipus determinat. Tot seguit presentem la manera com aquestes preguntes poden ser contestades a partir de la síntesi.

**Pot una transacció de tipus TTR canviar l'estat d'un o més objectes de la classe CO?** Aquesta pregunta té dues possibles respostes: "sí" i "no". Per contestar-la, cal saber si alguna de les classe d'esdeveniments estructurals induïdes potencialment pel tipus de transacció TTR pertany al conjunt de classes d'esdeveniments estructurals de la classe d'objectes CO.

La definició en lògica de primer ordre de la condició que s'ha de complir per dir que una transacció de tipus TTR pot canviar l'estat d'un o més objectes d'una classe CO és la següent:

$$\begin{aligned}&\forall\ TTR, CO\ (CO \in Objectes \wedge\\ &\quad \exists\ CEs\ (inducció(TTR,CEs) \wedge CEs \in EsdEstructurals(CO) \wedge\\ &\quad (CEs = Inserció(CO) \vee CEs = Esborrat(CO) \vee\\ &\quad \exists\ A\ (CEs \in Modificació(A) \wedge A \in Atributs(CO)))) \rightarrow\\ &\qquad\qquad potCanviarEstatObjectes(TTR,CO)).\end{aligned}$$

**Exemple 6.1.1**

Suposem que ens interessa saber si una transacció de tipus TTR = {nouSou} pot modificar l'estat d'un o més objectes de la classe *departament*. La resposta és "no". Això vol dir que, una transacció de tipus TTR mai no podrà inserir o esborrar objectes de la classe *departament* i tampoc no podrà modificar el valor d'algun dels atributs d'un *departament*. ♦

**A quines classes d'objectes pertanyen els objectes als quals pot afectar una transacció de tipus TTR, canviant-ne l'estat?** La resposta a aquesta pregunta és un conjunt de classes d'objectes. Per contestar-la, cal veure per a cadascuna de les classes d'objectes del model, si una transacció de tipus TTR pot canviar l'estat d'un o més dels seus objectes. Les classes d'objectes que compleixin aquesta condició han de formar part del conjunt resultat.

La definició en lògica de primer ordre de quines classes d'objectes poden veure modificat el seu estat a conseqüència d'una transacció de tipus TTR, és la següent:

$$\forall \text{ CO, TTR (CO} \in \text{ Objectes} \wedge \text{ potCanviarEstatObjectes(TTR,CO)} \rightarrow \text{CO} \in \text{ objectesQuePotModificar(TTR)).}$$

**Exemple 6.1.2**

Suposem que estem interessats a saber a quines classes d'objectes pertanyen els objectes que poden veure's afectats per una transacció de tipus TTR = {nouSou}. La resposta és el conjunt *{treballador, director}*. Això vol dir que una transacció de tipus TTR mai no podrà canviar l'estat d'objectes d'una classe que no sigui la classe *treballador* o la classe *director*. ♦

## 6.2 Preguntes sobre la modificació de l'estat d'atributs

A l'hora de validar un model conceptual pot ser interessant saber si una transacció d'un cert tipus pot modificar el valor d'un atribut d'un o més objectes de la classe per a la qual està definit. En aquesta secció veurem algunes preguntes que poden ajudar en aquest sentit.

**Pot una transacció de tipus TTR modificar el valor de l'atribut A d'algun objecte de la classe per a la qual A està definit?** Aquesta pregunta té dues possibles respostes: "sí" i "no". Per contestar-la, ens cal saber si alguna de les classes d'esdeveniments estructurals induïdes potencialment pel tipus de transacció TTR pertany al conjunt de classes d'esdeveniments estructurals de modificació de l'atribut A.

La definició en lògica de primer ordre de la condició que s'ha de complir per dir que una transacció de tipus TTR pot modificar el valor de l'atribut A d'un o més objectes de la classe per a la qual A està definit, és la següent:

$$\forall \text{ TTR, A, CO (CO} \in \text{ Objectes} \wedge \text{ A} \in \text{ Atributs(CO)} \wedge \exists \text{ CEs (inducció(TTR,CEs)} \wedge \text{ CEs} \in \text{ Modificació(A))} \rightarrow \text{potModificarAtribut(TTR,A)).}$$

**Exemple 6.2.1**

Suposem que estem interessats a saber si una transacció de tipus TTR = {nouSou, nouTreballador} pot modificar el valor de l'atribut *plantilla* d'algun objecte de la classe en què aquest atribut està definit. La resposta és "sí". Per tant, hi ha la possibilitat que una transacció de tipus TTR modifiqui el valor de l'atribut *plantilla.* ♦

**A quins atributs d'un model pot, una transacció de tipus TTR, modificar-ne el valor?** La resposta a aquesta pregunta és un conjunt d'atributs. Per contestar-la, cal veure per a cadascun dels atributs del model si una transacció de tipus TTR pot modificar el seu valor. Els atributs que compleixin aquesta condició han de formar part del conjunt resultat.

La definició en lògica de primer ordre dels atributs d'un model als quals una transacció de tipus TTR pot modificar el valor, és la següent:

$$\forall A, CO, TTR\ (CO \in Objectes \wedge A \in Atributs(CO) \wedge potModificarAtribut(TTR,A) \rightarrow A \in atributsQuePotModificar(TTR)).$$

**Exemple 6.2.2**

Suposem que estem interessats a saber els atributs del model exemple als quals pot, una transacció de tipus TTR = {nouSou}, modificar el valor. La resposta és el conjunt *{treballadorSou, directorSou, directorSobreSou}*. No hi ha cap altre atribut, tret dels que conté aquest conjunt, que puguin veure's afectats per una transacció del tipus *{nouSou}*. ♦

**A quins atributs d'una classe d'objectes CO pot, una transacció de tipus TTR, modificar-ne el valor?** La resposta a aquesta pregunta és un conjunt d'atributs de la classe d'objectes CO. Per contestar-la, cal veure per a cadascun dels atributs definits de la classe d'objectes CO, si una transacció de tipus TTR pot modificar el seu valor. Els atributs de la classe d'objectes CO que compleixin aquesta condició han de formar part del conjunt resultat.

La definició en lògica de primer ordre dels atributs de la classe d'objectes CO d'un model als quals una transacció de tipus TTR pot modificar el valor, és la següent:

$$\forall A, CO, TTR\ (CO \in Objectes \wedge A \in Atributs(CO) \wedge potModificarAtribut(TTR,A) \rightarrow A \in atributsObjQuePotModificar(TTR,CO)).$$

**Exemple 6.2.3**

Si preguntem a quins atributs de la classe *departament* pot, una transacció de tipus

TTR ={nouSou}, modificar el valor, la resposta és el conjunt *{}*. Això vol dir que no hi ha cap atribut de la classe *departament* que pugui modificar una transacció del tipus *{nouSou}*. ♦

## 6.3 Preguntes sobre la generació d'esdeveniments

Ara presentem algunes preguntes referents a la possible generació d'esdeveniments a conseqüència de l'ocurrència d'un cert tipus de transacció.

**Pot una transacció de tipus TTR generar esdeveniments de la classe G?** Aquesta pregunta té dues possibles respostes: "sí" i "no". Per contestar-la, cal saber si la classe d'esdeveniments estructurals de la classe d'esdeveniments generats G és induïda potencialment pel tipus de transacció TTR.

La definició en lògica de primer ordre de la condició que s'ha de complir per dir que una transacció de tipus TTR pot generar esdeveniments de la classe d'esdeveniments G, és la següent:

$$\forall \text{ TTR, G, CO (CO} \in \text{ Objectes } \wedge \text{ G} \in \text{ Generacions(CO)} \wedge$$
$$\exists \text{ CEs (inducció(TTR,CEs) } \wedge \text{ CEs = Generació(G))} \rightarrow$$
$$\text{potGenerarEsdeveniments(TTR,G)).}$$

**Exemple 6.3.1**

Suposem que estem interessats a saber si una transacció de tipus TTR = {nouDirector} pot generar esdeveniments de la classe *important*. La resposta és "no". Per tant, en cap cas una transacció de tipus TTR podrà generar esdeveniments de la classe *important*. ♦

**De quines classes d'esdeveniments generats una transacció de tipus TTR pot generar esdeveniments?** La resposta a aquesta pregunta és un conjunt de classes d'esdeveniments generats. Per contestar-la, cal veure per a cadascuna de les classes d'esdeveniments generats del model, si una transacció de tipus TTR pot generar-ne algun esdeveniment. Les classes d'esdeveniments generats que compleixin aquesta condició han de formar part del conjunt resultat.

La definició en lògica de primer ordre de quines classes d'esdeveniments generats, una transacció de tipus TTR pot generar esdeveniments, és la següent:

$$\forall \text{ G, CO, TTR (CO} \in \text{ Objectes } \wedge \text{ G} \in \text{ Generacions(CO)} \wedge$$
$$\text{potGenerarEsdeveniments(TTR,G)} \rightarrow$$
$$\text{G} \in \text{ esdevQuePotGenerar(TTR)).}$$

**Exemple 6.3.2**

Tornem al cas d'una transacció de tipus TTR = {nouDirector}. Si preguntem de quines classes d'esdeveniments generats, una transacció de tipus TTR pot generar esdeveniments, la resposta és el conjunt *{}*. Això vol dir que una transacció del tipus *{nouDirector}* no pot generar esdeveniments de cap classe. ♦

**De quines classes d'esdeveniments generats d'una classe d'objectes CO, una transacció de tipus TTR pot generar esdeveniments?** La resposta a aquesta pregunta és un conjunt de classes d'esdeveniments generats de la classe d'objectes CO. Per contestar-la, cal veure per a cadascuna de les classes d'esdeveniments generats de la classe d'objectes CO, si una transacció de tipus TTR pot generar-ne esdeveniments. Les classes d'esdeveniments generats definits per a la classe d'objectes CO que compleixin aquesta condició han de formar part del conjunt resultat.

La definició en lògica de primer ordre de quines classes d'esdeveniments generats d'una classe d'objectes CO, una transacció de tipus TTR pot generar esdeveniments, és la següent:

$$\forall\, G, CO, TTR\ (CO \in Objectes \wedge G \in Generacions(CO) \wedge potGenerarEsdeveniments(TTR,G) \rightarrow G \in esdevObjQuePotGenerar(TTR,CO)).$$

**Exemple 6.3.3**

Suposem ara un tipus de transacció TTR = {nouDirector,nouSou}, i suposem que volem saber de quines classes d'esdeveniments generats de la classe d'objectes *treballador,* pot aquest tipus de transacció generar esdeveniments. La resposta és el conjunt *{important}*. ♦

## 6.4 Preguntes sobre la violació de restriccions d'integritat

Altres preguntes útils són les que permeten saber les relacions entre l'ocurrència d'un tipus de transacció i la violació de restriccions d'integritat. Són les següents.

**Pot una transacció de tipus TTR, violar la restricció d'integritat Ic?** Aquesta pregunta té dues possibles respostes: "sí" i "no". Per contestar-la, ens cal saber si la classe d'esdeveniments estructurals de violació de la restricció d'integritat Ic és induïda potencialment pel tipus de transacció TTR.

La definició en lògica de primer ordre de la condició que s'ha de complir per dir que una transacció de tipus TTR pot violar la restricció d'integritat Ic, és la següent:

$$\forall \text{ TTR, Ic, CO } (\text{CO} \in \text{Objectes} \wedge \text{Ic} \in \text{Restriccions(CO)} \wedge \exists \text{ CEs } (\text{inducció(TTR,CEs)} \wedge \text{CEs} = \text{Restricció(Ic)}) \rightarrow \text{potViolarRestricció(TTR,Ic)}).$$

**Exemple 6.4.1**

Suposem que volem saber si una transacció de tipus TTR = {nouSou, nouDirector} pot violar la restricció d'integritat *plantillaExcedida*. La resposta és "no". Així, quan ocorri una transacció de tipus TTR, podrem assegurar que mai no es violarà la restricció d'integritat *plantillaExcedida*. ♦

**Una transacció de tipus TTR, quines restriccions d'integritat d'un model pot violar?** La resposta a aquesta pregunta és un conjunt de restriccions d'integritat. Per contestar-la, cal veure per a cadascuna de les restriccions d'integritat del model, si una transacció de tipus TTR pot violar-la. Les restriccions d'integritat que compleixin aquesta condició han de formar part del conjunt resultat.

La definició en lògica de primer ordre de quines són les restriccions d'integritat d'un model que una transacció de tipus TTR pot violar, és la següent:

$$\forall \text{ Ic, CO, TTR } (\text{CO} \in \text{Objectes} \wedge \text{Ic} \in \text{Restriccions(CO)} \wedge \text{potViolarRestricció(TTR,Ic)} \rightarrow \text{Ic} \in \text{restriccióQuePotViolar(TTR)}).$$

**Exemple 6.4.2**

Tornant al cas de transaccions del tipus TTR = {nouSou, nouDirector}, si preguntem quines són les restriccions definides en el model que una transacció d'aquest tipus pot violar, la resposta és *{ic4, noTreballaA, ic5, ic6, ésElDirector}*. En cap cas una transacció de tipus TTR podrà violar restriccions d'integritat que no estiguin en aquest conjunt. ♦

**Una transacció de tipus TTR, quines restriccions d'integritat definides per una classe d'objectes CO pot violar?** La resposta a aquesta pregunta és un conjunt de restriccions d'integritat definides per una classe d'objectes CO. Per contestar-la, cal veure, per a cadascuna de les restriccions d'integritat de CO, si una transacció de tipus TTR la pot violar. Les restriccions d'integritat definides per la classe d'objectes CO que compleixin aquesta condició han de formar part del conjunt resultat.

La definició en lògica de primer ordre de quines són les restriccions d'integritat definides per la classe d'objectes CO que una transacció de tipus TTR pot violar, és la següent:

$$\forall \text{ Ic, CO, TTR } (\text{CO} \in \text{Objectes} \wedge \text{Ic} \in \text{Restriccions(CO)} \wedge \text{potViolarRestricció(TTR,Ic)} \rightarrow \text{Ic} \in \text{restricciónObjQuePotViolar(TTR,CO)}).$$

**Exemple 6.4.3**

Tornant al mateix tipus de transacció usat a l'exemple anterior, suposem que estem interessats a saber quines restriccions definides a la classe d'objectes *departament* pot violar. La resposta és *{}*. Això vol dir que mai una transacció de tipus *{nouSou,nouDirector}* podrà violar una de les restriccions d'integritat definides per a la classe *departament*. ♦

## 6.5 Preguntes sobre canvis d'estat concrets

Fins ara les preguntes feien referència als elements d'un model conceptual. En aquesta secció, baixarem de nivell i proposarem preguntes sobre canvis d'estat concrets. És a dir que, per exemple, no preguntarem si els objectes d'una classe poden canviar a causa d'una transacció d'un cert tipus, sinó si una transacció d'un cert tipus pot causar la inserció d'objectes en una classe. Per poder fer aquest tipus de preguntes, la persona encarregada de la validació del model haurà de conèixer les classes d'esdeveniments estructurals que s'han deduït del model en fer la síntesi.

**Pot, una transacció de tipus TTR, causar un o més canvis representats per la classe d'esdeveniments estructurals CEs?** Aquesta pregunta té dues possibles respostes: "sí" i "no". Per respondre, cal saber si CEs és induïda potencialment per TTR.

La definició en lògica de primer ordre de la condició que s'ha de complir per dir que una transacció de tipus TTR pot causar un o més canvis dels representats per la classe d'esdeveniments estructurals CEs, és la següent:

$$\forall \text{ TTR, CEs, CO } (\text{CO} \in \text{Objectes} \wedge \text{CEs} \in \text{EsdEstructurals(CO)} \wedge \text{inducció(TTR,CEs)} \rightarrow \text{potProvocarCanvis(TTR,CEs)}).$$

**Exemple 6.5.1**

Com a exemple, suposem que volem saber si una transacció de tipus *{nouTreballador}* pot causar la inserció d'un o més objectes a la classe d'objectes *treballador*, canvi representat per la classe d'esdeveniments estructurals *i_treballador*. La resposta és "sí", ja que aquesta classe d'esdeveniments estructurals és induïda potencialment per *{nouTreballador}*. ♦

**Quines classes d'esdeveniments externs fan falta en un tipus de transacció, per tal que les transaccions del tipus puguin causar els canvis representats per la classe d'esdeveniments estructurals CEs?** Com que pot haver-hi més d'un conjunt de classes d'esdeveniments externs que compleixi aquesta condició, la resposta a aquesta pregunta serà un conjunt de conjunts. Cada element en el conjunts resposta contindrà les mínimes classes d'esdeveniments externs necessàries per causar potencialment esdeveniments de la classe CEs.

La definició en lògica de primer ordre de quins conjunts de classes d'esdeveniments externs del model poden constituir tipus de transaccions que puguin causar els canvis representats per una classe d'esdeveniments estructurals CEs, és la següent:

$$\forall\ ConjN, CEs$$
$$(ConjN \subseteq EsdExterns \wedge potProvocarCanvis(ConjN,CEs) \wedge$$
$$\neg\ \exists\ ConjA\ (ConjA \subset EsdExterns \wedge potProvocarCanvis(ConjA,CEs) \wedge$$
$$ConjA \subset ConjN) \rightarrow$$
$$ConjN \in conjuntsClassesNecessàriesTTR(CEs)).$$

**Exemple 6.5.2**

Si preguntem quines classes d'esdeveniments externs fan falta per causar els canvis representats per la classe d'esdeveniments estructurals *i_treballaA*, és a dir per donar un nou valor a l'atribut *treballaA* d'un o més objectes de la classe *treballador*, la resposta és *{{nouTreballador}, {canviDepartament}}*. Així, si volem que un tipus de transacció provoqui que s'insereixi un nou valor a l'atribut *treballaA* d'un o més objectes de la classe *treballador*, ens cal un tipus de transacció amb esdeveniments de la classe *nouTreballador* o amb esdeveniments de la classe *canviDepartament*. ♦

**És possible que, tenint en compte el model, ocorrin un o més canvis representats per la classe d'esdeveniments estructurals CEs?** Aquesta pregunta té dues possibles respostes: "sí" i "no". La resposta és "sí" en el cas que existeixi una transacció de tipus TTR que pugui causar els canvis representats per la classe d'esdeveniments estructurals CEs. En el cas contrari, podrem afirmar que no hi ha manera que es puguin arribar a causar esdeveniments de la classe CEs.

La definició en lògica de primer ordre de la condició que s'ha de complir per dir que donat un model poden ocórrer els canvis representats per una classe d'esdeveniments estructurals CEs, és la següent:

$$\forall\ CEs, ConjN\ (ConjN \in conjuntsClassesNecessàriesTTR(CEs) \rightarrow$$
$$sónPossiblesCanvis(CEs)).$$

**Quines classes d'esdeveniments externs fan falta afegir a un tipus de transacció TTR perquè les transaccions del tipus puguin causar un o més canvis representats per la classe d'esdeveniments estructurals CEs?** Com que pot haver-hi més d'un conjunt de classes d'esdeveniments externs que compleixi aquesta condició, la resposta que donarem a aquesta pregunta és un conjunt de conjunts. Cada element en el conjunt resposta agruparà les classes d'esdeveniments mínimes necessàries per tal que, afegides a un tipus de transacció TTR, conformin un nou tipus de transacció que puguin causar esdeveniments de la classe CEs.

La definició en lògica de primer ordre de quins conjunts de classes d'esdeveniments externs del model poden constituir, en ser afegits a un tipus de transacció TTR, tipus de transaccions que puguin causar els canvis representats per una classe d'esdeveniments CEs, és la següent:

$$
\begin{aligned}
&\forall\ \text{ConjN, CEs, TTR, ConjR} \\
&\quad (\text{ConjN} \subseteq \text{EsdExterns} \wedge \text{potProvocarCanvis(ConjN,CEs)} \wedge \\
&\quad \text{TTR} \subseteq \text{ConjN} \wedge \text{ConjR} = \text{ConjN} - \text{TTR} \wedge \\
&\quad \neg\, \exists\ \text{ConjA (ConjA} \subset \text{EsdExterns} \wedge \text{potProvocarCanvis(ConjA,CEs)} \wedge \\
&\qquad \text{ConjA} \subset \text{ConjN} \wedge \text{TTR} \subseteq \text{ConjA}) \rightarrow \\
&\qquad \text{ConjR} \in \text{conjuntsClassesNecessàriesAfegirTTR(TTR,CEs)).}
\end{aligned}
$$

**Exemple 6.5.3**

Suposem que volem saber quines classes d'esdeveniments externs fan falta afegir a un tipus de transacció *{nouDepartament, nouSou}* perquè les transaccions del tipus puguin causar esdeveniments de la classe d'esdeveniments estructurals *i_responsable*, és a dir, perquè puguin causar insercions d'un valor a l'atribut *responsable* d'un o més objectes de la classe *departament*. La resposta és *{{nouDirector}}*. Per tant, l'única solució és afegir-hi esdeveniments de la classe *nouDirector*. ♦

# 7. Conclusions i recerca futura

## 7.1 Conclusions

Els llenguatges amb interacció implícita semblen més adequats per a l'especificació de sistemes d'informació que els que presenten interacció explícita, ja que no obliguen en l'etapa d'especificació de requeriments d'un sistema a prendre decisions pròpies de l'etapa de disseny.

Com a contrapartida, els models escrits amb un llenguatge amb interacció implícita són més difícils de validar. La raó és que en aquests models és difícil de veure quin pot ser l'efecte d'un o més esdeveniments que es comuniquin al sistema en un instant determinat, i per tant, de comprovar si el model es correspon realment als requeriments de l'usuari.

El determinar Esquemes d'Interacció en un model amb interaccions implícites és una de les maneres d'ajudar al dissenyador en la validació del model. En aquesta tesi hem proposat un mètode per a la determinació d'Esquemes d'Interacció en models orientats a objectes amb interacció implícita.

D'altra banda, hem classificat el mètode com una tècnica de validació de models conceptuals de facilitació del raonament. I dins d'aquest grup, com una tècnica d'inspecció de models.

Hem exposat l'avantatge que suposa el que es tracti d'un mètode que proporciona informació sobre l'aspecte dinàmic d'un model sense agafar com a punt de partida un estat concret del sistema i una transacció o conjunt d'esdeveniments externs concrets.

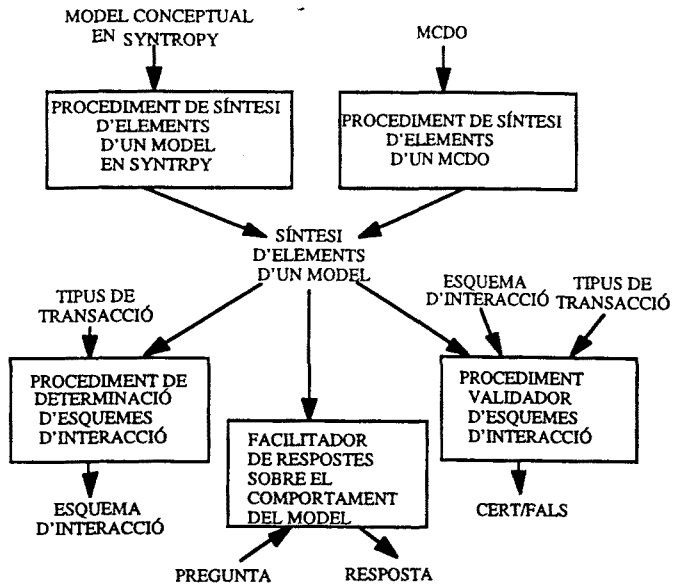


**Figura 7.1.1**

També hem vist que és un mètode general, que podrà ser aplicat a qualsevol model escrit en un llenguatge orientat a objectes amb interacció implícita, que sigui suficientment formal, simplement definint com fer la síntesi d'elements d'un model en aquest llenguatge.

Hem descrit el procediment semiformal de síntesi d'elements d'un model a partir de models en Syntropy, i de MCDO.

Hem proposat un procediment formal de determinació d'Esquemes d'Interacció vàlids (figura 7.1.1), que, a partir de la síntesi dels elements d'un model i un tipus de transacció, dóna com a resultat un Esquema d'Interacció vàlid per al processament del tipus de transacció. D'aquest procediment n'hem adjuntat la seva implementació en Prolog.

El procediment anterior dóna com a resultat un únic Esquema d'Interacció vàlid de tots els possibles per a un tipus de transacció.

En aquesta tesi, hem donat la definició de quan un Esquema d'Interacció és vàlid per a un tipus de transacció, i hem proposat un procediment validador (figura 7.1.1), que, a partir de la síntesi dels elements d'un model, un tipus de transacció i un Esquema

d'Interacció, ens respon si l'Esquema d'Interacció és vàlid per al processament del tipus de transacció. D'aquest procediment n'hem fet també la implementació en Prolog.

La síntesi d'elements d'un model pot ser utilitzada per obtenir documentació complementària sobre un model a base de formular preguntes que poden ser contestades a partir de la síntesi. En la tesi, hem vist una mostra de les preguntes que es poden contestar. El facilitador de respostes que apareix a la figura 7.1.1 representa el component per respondre a aquestes preguntes.

Finalment, hem vist que els antecedents en el camp de la validació de models conceptuals orientats a objectes té poc a veure amb el nostre mètode i que els antecedents més relacionats amb el nostre mètode són els treballs que existeixen d'anàlisi de conjunts de regles.

## 7.2 Recerca futura

A continuació exposem algunes línies de recerca de treball futur.

Una primera línia de recerca és ampliar el nostre mètode perquè pugui ser aplicat a models escrits en llenguatge ROSES [CBC+96,CSO+97]. Això suposa veure com ha de ser el procés de síntesi d'elements en un model en aquest llenguatge.

L'aspecte dinàmic d'un sistema es modela en ROSES mitjançant un conjunt de regles independents, cadascuna de les quals dóna una relació entre una causa i un efecte. En ROSES l'efecte que pot tenir un o més esdeveniments que ocorrin en un cert instant sobre l'estat del sistema és difícil de saber, a l'igual que les interaccions que poden provocar. El mètode per determinar Esquemes d'Interacció podria ajudar a validar un model en aquest llenguatge.

Una segona línia de recerca és ampliar el nostre mètode per tal que pugui ser aplicat a models en Syntropy i MCDO que aquí no hem considerat per com modelaven el comportament o aspecte dinàmic d'un sistema.

En Syntropy no hem considerat aquells models en què un esdeveniment extern pugui provocar, de manera directa o indirecta, més d'un esdeveniment d'un mateix tipus amb un cert ordre d'ocurrència entre ells. També hem descartat aquells models en què un esdeveniment extern pugui provocar, de manera directa o indirecta, canvis en una part concreta de l'estat del sistema, causats per esdeveniments de diferent tipus amb un cert ordre d'ocurrència entre ells.

En MCDO no hem considerat models amb regles de deducció que presentin recursivitat directa o indirecta.

Si apliquem el nostre mètode, tal com està, a una síntesi d'elements d'un d'aquests models hi haurien casos en que el procediment de determinació d'Esquemes no donaria un resultat correcte. Això ve del fet que, en aquests models, la definició dels conceptes de classe d'esdeveniments computada i classe d'esdeveniments parcialment computada, en que es basa el nostre mètode, no és correcta. De fet, en aquests models mai no es podrà assegurar, sinó es tracta d'una transacció concreta i un estat del sistema concret, quan una classe d'esdeveniments està computada o parcialment computada, ja que una regla d'esdeveniments no se sap les vegades que s'haurà d'aplicar per arribar a computar tots els esdeveniments estructurals, de la classe a què la regla es refereix.

Pensem que la solució pot venir de considerar possibles, Esquemes d'Interacció on un tipus d'interacció pugui aparèixer diverses vegades dins l'Esquema assignat a diferents estats. Al procediment de determinació d'Esquemes se li hauria de canviar la condició d'acabament, probablement fent que acabés quan no trobés cap tipus d'interacció a afegir a l'Esquema, o bé quan reconeixes un grup d'un o més tipus d'interacció en l'Esquema que es repetís varies vegades sense altres tipus d'interacció entre mig.

Una tercera línia de recerca és refinar el mètode, per tal de donar resultats una mica més concrets que ara. Sembla que això pot ser possible ampliant la definició de regla d'esdeveniments.

Una regla d'esdeveniments està definida mitjançant tres funcions: Regles(CEs) que representa el conjunt de regles que es refereixen a la classe d'esdeveniments estructurals CEs, Causa(R) que representa el conjunt de classes d'esdeveniments causa de la regla R i Prerequisit(R) que representa el conjunt de classes d'esdeveniments prerequisit de la regla R.

Si s'afegís a aquesta definició els atributs de les classes d'esdeveniments estructurals i externs de les classes d'esdeveniments estructurals, sembla que seria possible, en alguns casos, determinar Esquemes d'Interacció més simplificats.

Per exemple, suposem el cas d'una regla d'esdeveniments $r$ que es refereix a una classe d'esdeveniments estructurals $s(X)$ amb un conjunt de causes $\{p(X),q(X)\}$, on $p$ i $q$ són classes d'esdeveniments externs, i suposem que tenim un tipus de transacció TTR = {p(A),q(B)}, amb A diferent de B. Amb el mètode ampliat, podríem assegurar

que donat un tipus de transacció TTR mai poden produir-se esdeveniments de la classe d'esdeveniments estructurals *s* a través de la regla *r*.

Finalment, una quarta línia de recerca és millorar el procediment de determinació d'Esquemes d'Interacció. Com hem dit, aquest procediment dóna com a resultat un únic Esquema d'Interacció vàlid d'entre tots els possibles per a un tipus de transacció. Una possible millora d'aquest procediment és que passi a donar tots els Esquemes d'Interacció vàlids.

Tenint el compte que per a un tipus de transacció hi poden haver gran quantitat d'Esquemes d'Interacció vàlids, el nou procediment podria donar la possibilitat de fixar requeriments sobre els Esquemes d'Interacció que es vol obtenir. Aquests requeriments podrien ser, per exemple, el nombre d'estats en que es vol distribuïts els tipus d'interaccions, la precedència que es vol mantenir entre dos tipus d'interaccions,...

# Referències

[AHW95] Aiken,A.; Hellerstein,J.M.; Widom,J. Static Analysis Techniques for Predicting the Behaviour of Active Database Rules. *ACM Transactions on Database Systems*, 20(1), Març 1995, pàg. 3-41.

[BCC+96] Barceló,M.; Costa,P.; Costal,D.; Olivé,A.; Quer,C.; Roselló,A.; Sancho,M.R. El llenguatge ROSES. Part I. *Report tècnic LSI-96-55-R*, Universitat Politècnica de Catalunya. Barcelona, Setembre 1996.

[BCP95] Baralis,E.;Ceri,S.;Paraboschi,S. Improved Rule Analysis by Means of Triggering and Activation Graphs. *Proceedings of DOOD'95*. Singapore. Desembre 1995.

[BMM90] Bry,F.;Manthey,R.;Martens,B. Integrity verification in knowledge bases. *RCLP 1990/1991*. Pàg. 114-139.

[Bub86] Bubenko,J.A. Information system methodologies - A research view. *A [OSV86]*. Pàg. 289-318.

[Bub88] Bubenko,J.A. Selecting a Strategy for Computer-Aided Software Engineering (CASE). *SYSLAB Report.* N.59. University of Stockolm, Juny 1988.

[CAB+94] Coleman,D.; Arnold,P.; Bodoff,S.; Dollin,C.; Gilchrist,H.; Hayes,F.; Jeremes,P. *Object-Oriented Development. The Fusion Method.* Prentice Hall, 1994.

[CBC+96] Costa,P.; Barceló,M.; Costal,D.; Olivé,A.; Quer,C.; Roselló,A.; Sancho,M.R. Las clases de objetos en ROSES. *Proceedings de las Primeras Jornadas de investigación y docencia en Bases de Datos*. La Corunya, Juny 1996.

[CeW91] Ceri,S.; Widom,J. Deriving production rules for incremental view maintenance. *Proceedings of VLDB'91*. Barcelona, 1991, pàg. 577-589.

[CoD94] Cook,S.; Daniels,J. *Designing Object Systems. Object Oriented Modelling with Syntropy*. The Object-Oriented Series. Prentice Hall, 1994.

[Cos95] Costal,D. *Un mètode de planificació basat en l'actualització de vistes en bases de dades deductives*. Tesi doctoral. Universitat Politècnica de Catalunya, 1995.

[CoY91] Coad,P.; Yourdon,E. *Object-Oriented Analysis*. Yourdon Press, 1991

[CSO+97] Costal,D.; Sancho,M.R.; Olivé,A.; Barceló,M.; Costa,P.; Quer,C.; Roselló,A. The Cause-Effect Rules of ROSES. *Proceedings of the First East-European Symposium on Advances in Databases and Information Systems (ADBIS'97)*, St. Petersburg, Setembre 1997, pàg. 399-405.

[CTU+96] Costal,D.; Teniente,E.; Urpí,T.; Farré,C. Handling Conceptual Model Validation by Planning. *Proceedings of CAiSE'95*, Crete, June 1995.

[DaJ97] Dalianis,H.;Johannesson,P. Explaining Conceptual Models - An Architecture and Design Principles. *Proceedings of the Entity-Relationship'97*, Los Angeles, 1997.

[Dal92] Dalianis,H. A Method for Validating a Conceptual Model by Natural Language. *Proceedings of CAiSE'92*, Manchester, UK, Maig 1992, pp. 425,444.

[Dav90] Davis,A.M. *Software Requirements: analysis and specification*. Prentice-Hall, 1990.

[DeG94] Deiters,S.; Griefahn,U. Propagation Rule Compiler: Tool Specification. *Technical Report IDEA.DE.22.O.001*, Novembre, 1994.

[DFv96] Derksen,C.F.; Frederiks,P.J.M.; van der Weide,Th.P. Paraphrasing as a Technique to Support Object-Oriented Analysis. *Proceedigns of the Second Workshop on Application of Natural Language to Databases (NLDB'96).* Amsterdam. Juny, 1996.

[Gri82] van Griethuysen,J.J. (Ed.). Concepts and terminology for the conceptual schema and the information base. *ISO/TC97/SC5/WG3,* Març 1982.

[GrM94] Griefahn,U.; Manthey,R. Update Propagation in Chimera, an Active DOOD Language. *Proceedings of the 5th International Workshop on the Deductive Approach to Information Systems and Databases,* Aiguablava, 1994.

[GrK97] Grau,A.; Kowsari,M. A Validation System for Object-Oriented Specifications of Information Systems. *Proceedings of the First East-European Symposium on Advances in Databases and Information Systems (ADBIS'97),* St. Petersburg, Setembre 1997.

[Gul93] Gulla,J.A. *Explanation Generation in Information Systems Engineering.* PhD Thesis. IDT, NTH, Trondheim, September 1993.

[Gul96] Gulla,J.A. A General Explanation Component for Conceptual Modelling in CASE Environments. *ACM Transactions on Information Systems,* 14(3), Juliol 1996.

[Küc91] Küchenchoff,V. On the efficient computation of the difference between consecutive database states, *Proceedigns of the DOOD'91,* Springer-Verlag, Munich, 1991, pàg. 478-502.

[Jac92] Jacobson,I. Object-Oriented Software Engineering. A Use Case Driven Approach. Addison Wesley, 1992.

[JSH+96] Jungclaus,R.; Saake,G.; Hartmann,T.; Sernadas,C. TROLL - A Language for Object-Oriented Specification of Information Systems. *ACM Transactions on Information Systems,* 14(2), 1996, pàg. 175-211.

[Lar98] Larman,C. *Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design.* Prentice Hall, 1998.

[LeL96] Lee,S.Y.; Ling,T.W. Further Improvement on Integrity Constraint Checking for Stratifiable Deductive Databases. *Proceedings of the VLDB'96*, Munbai (Bombay), India, 1996.

[LeL98] Lee,S.Y.; Ling,T.W. A Path Removing Technique for Detecting Trigger Termination. *Proceedings of the EDBT'98*, Valencia, Spain, 1998.

[LiK93] Lindland,O.I.; Krogstie,J. Validating Conceptual Models by Transformational Prototyping. *Proceedings of CAiSE'93*, Paris, Juny 1993.

[Lin93] Lindland,O.I. *A prototyping approach to validation of conceptual models in information systems engineering*. PhD Thesis. IDT, NTH, Trondheim, May 1993.

[LIT84] Lloyd,J.W.; Topor,R.W. Making Prolog more expressive. *Journal of Logic Programming*, N.3, 1984, pàg. 225-240.

[Llo87] Lloyd,J.W. *Foundations of logic programming(2nd Ed.)*, Springer-Verlag, 1987.

[LTP91] Loucopoulos,P.; Theodoulidis,B.; Pantazis,D. Business rules modelling: conceptual modelling and object specifications. *A [VMR91]*. Pàg. 323-342.

[MaO92] Martin,J.; Odell,J. *Object-Oriented Analysis & Design*. Prentice Hall. 1992.

[PaW97] Parsons,J.; Wand,Y. Using Objects for Systems Analysis. *Communications of the ACM*, 40(12), Decembre 1997, pàg. 104-110.

[Oli86] Olivé,A. On the design and implementation of information systems form deductive conceptual models. *Proceedings of the VLDB'89*. Amsterdam, 1989, pàg. 3-11.

[Oli89] Olivé,A. A comparison of the operational and deductive approaches to conceptual information systems modelling. *Proceedings of the IFIP'86*, North-Holland, Dublin, 1986, pàg. 91-96.

[OSV86] Olle,T.W.; Sol,H.G.; Verrijn-Stuart, A.A. (Eds) *Information systems design methodologies: Improving the practice*. North-Holland, 1986.

[Que91] Quer,C. Combining the object-oriented and the deductive approach for conceptual modelling. *Proceedings of the 3th International Workshop on the Deductive Approach to Information Systems and Databases (DAISD'91)*, Roses, 1991.

[Que94] Quer,C. An Execution Model for Change Computation in Deductive Databases. *Proceedings of the 5th International Workshop on the Deductive Approach to Information Systems and Databases (DAISD'94)*, Aiguablava, 1994.

[QuO93] Quer,C.; Olivé,A. Object Interaction in Object Oriented Deductive Conceptual Models. *Proceedings of CAiSE'93*, Paris, 1993, pàg. 374-396.

[QuO94] Quer,C.; Olivé,A. Determining Object Interaction in Object Oriented Deductive Conceptual Models. *Information Systems*, 19(3), 1994, pàg. 211-227.

[RBP+91] Rumbaugh,J.; Blaha,M; Premerlani,W.; Eddy,F.; Lorensen,W. *Object Oriented Modelling and Design*. Prentice Hall, 1991.

[RCB+89] Rosenthal,A.; Chakravarthy,S.; Blaustein,B.; Blakeley,J. Situation monitoring for active databases. Proceedings of the VLDB'89. Amstermdam, 1989, pàg. 455-464.

[RoP92] Rolland,C.; Proix,C. A Natural Language Approach for Requirements Engineeging. *Proceedings of CAiSE'92*, Manchester, UK, May 1992, pp. 257-277.

[San93] Sancho,M.R. Explaining the Behaviour of a Deductive Conceptual Model. *Proceedings of the 4th International Workshop on the Deductive Approach to Information Systems and Databases (DAISD'93)*, Lloret de Mar, 1993.

[San94] Sancho,M.R. *Disseny de transaccions a partir de models conceptuals deductius*. Tesi doctoral. UPC, 1994.



[Sel94] Seltveit,A.H. *Complexity Reduction in Information Systems Modelling*. PhD Thesis. IDT, NTH, Trondheim, December 1994.

[ShM92] Shlaer,S.; Mellor,S.J. *Object Lifecycles: Modeling the World in States.* Yourdon Press, Englewood Cliffs, NJ, 1992.

[Sis87] Sistac,J. *Construcció automàtica de prototipus de sistemes d'informació a partir de Models Conceptuals Deductius descrits amb llenguatge DADES* Tesi doctoral. UPC, 1987.

[Sis92] Sistac,J. The DADES/GP approach to automatic generation of IS prtotypes from a DCM. *Information Systems,* 17(2), 1992, pàg.195-208.

[UrO92] Urpí,T.; Olivé,A. A Method for Change Computation in Deductive Databases. *Proceedings of the VLDB'92,* Vancouver, Canada, 1992, pàg. 225-237.

[VMR91] Van Assche,F.; Moulin,B.; Rolland,C. (Eds) *Object oriented approach in information systems,* North-Holland, 1991.

[WiC96] Widom,J.; Ceri,S. *Active Database Systems.* Morgan Kaufmann Publishers. San Francisco, 1996.