System ::= SystemID {[Game]+} | Game}

SystemID ::= Identifier

Game ::= GameID { Composition [Rule]* [Interaction]+ }

GameID ::= Identifier

Composition ::= Turns [RoleList]? Participants, [Player]+ [Store]* [TransForce]* [BackTrack]*

Turns ::= { turns, TurnSize, Ordering, [MaxTurns]? }

TurnSize ::= magnitude: Number | single | multiple

Ordering ::= ordering: strict | liberal

MaxTurns ::= max: Number | RunTimeVar

RunTimeVar ::= $Identifier$

RoleList ::= { roles, { Role [, Role]+ } }

Role ::= speaker | listener | Identifier

Participants ::= { players, min: Number, max: Number | undefined }

Player ::= { player, id: PlayerID | RunTimeVar [, roles: { Roles }]? }

PlayerID ::= Identifier

Store ::= { store, id: StoreName, owner: StoreOwner, StoreStructure, Visibility, InitialContents }

StoreName ::= Identifier

StoreOwner ::= PlayerID | { PlayerID [, PlayerID] } | shared

StoreStructure::= structure:set | queue | stack

Visibility ::= visibility:public | private

InitialContents ::= contents: Content | RunTimeVar

TransForce ::= {transforce, {Locution[, Locution]?}, {Locution[, Locution]?}, [ForceID, Target]+ [, Requirements]?};

ForceID ::= Identifier

Target ::= Content | { SchemeApp, SchemeID }

BackTrack ::= {backtrack, on | off }

Rule ::= { rule, RuleID, scope:initial | turnwise | movewise, RuleBody }

RuleID ::= Identifier

RuleBody ::= Effects | Conditional [& Conditional]*

Effects ::= { Effect [& Effect]* }

Effect ::= EffectID( Parameter [, Parameter]* )

Parameter ::= Identifier | Number | Commitment | SystemID | GameID | PlayerID | MaxTurns | StoreName | StoreOwner | Requirements | Role | RunTimeVar | Condition | Effect | Content | Locution | Argument | Interaction | SchemeID | {Parameter[, Parameter]*}

Interaction ::= { interaction, MoveID, [, Role]? [, Target]? [,ForceID ,Target]? [,Opener]? ,RuleBody }

MoveID ::= Identifier

Opener ::= String

Conditional ::= { if Requirements then Effects [elseif Requirements then Effects]* [else Effects]? }

Requirements ::= { Condition [& Condition]* } | { Requirements [ || Requirements]* }

Condition ::= ConditionID( Parameter [, Parameter]* )

ConditionID ::= Identifier

Commitment ::= Content | Locution | Argument

Locution ::= < MoveID, Content >

Content ::= ContentSet | ContentVar | { ContentSet | ContentVar [,ContentSet | ContentVar]* }

ContentSet ::= UpperChar

ContentVar ::= LowerChar | !LowerChar

SchemeApp ::= <Content, Content>

Argument ::= < Conclusion, Premises >

Premises ::= { ContentVar [, ContentVar]* }

Conclusion ::= ContentVar

SchemeID ::= Identifier

Identifier ::= UpperChar [ UpperChar | LowerChar | Number]+

String ::= “[UpperChar | LowerChar | Number | Symbol]+”

Number ::= [0--9]+

UpperChar ::= [A--Z]+

LowerChar ::= [a--z]+

Symbol ::= ` ' | `?' | `,' | `.'

% Condition Predicates

RespondsTo ::= responds(MoveID [,Content ]? [, PlayerID | Role ]?)

Event ::= event( last | !last | past | !past, MoveID [,Content ]? [, PlayerID | Role ]? [, Requirements]?)

StoreInspection ::= inspect(in | !in | on | !on | top | !top, Commitment, StoreName, [PlayerID | Role ]? [, initial | past | current ]? )

RoleInspection ::= inrole( PlayerID, Role )

Magnitude ::= size( StoreName | LegalMoves, PlayerID, empty | !empty | Number )

StoreComparison ::= magnitude( StoreName, PlayerID | Role, greater | smaller | equal | !equal, StoreName, PlayerID | Role, )

DialogueSize ::= numturns( SystemName, Number | RunTimeVar)

Correspondence ::= corresponds( Argument, SchemeID)

Relation ::= relation(Content | Argument, backing | warrant, Content | Argument )

CurrentPlayer ::= player(PlayerID | Role)

ExternalCondition::= extCondition( Identifier, {Parameter [, Parameter]*} )

% Effect Predicates

Move ::= move(add | delete, next | future, MoveID, [, Content ]? [, PlayerID | Role ]? [, Requirements ]? )

StoreOp ::= store( add | remove, Commitment, StoreName, PlayerID | Role [, Requirements ]?)

StatusUpdate ::= status( active | inactive | complete | incomplete | initiate | terminate, SystemID | GameID [, Requirements ]?, [Effect]?)

RoleAssignment::= assign( PlayerID | Role, Role [, Requirements ]? )

ExternalEffect::= extEffect(Identifier [, Identifier ]*)