Information Quality Policy Specification Language

In this document, first we introduce our Information Quality (IQ) policy specification language, and then we define the rules that enable for the automatic derivation of such policies from the requirements model.

IQ Policy Specification Language

Our IQ policy specification language provides a clear way for specifying IQ requirements in terms of IQ policies. The language supports three types of polices namely: permit, forbid, and obligate policies that are used to control four different types of activities over information, namely, produce, read, modify, and send. The IQ policy specification language can be represented in BNF notation [1] as:

IQ_Policy = (**Permit** | **Forbid** | **Obligate**)

IQ_Policy = **IQ_Policy_name(**Actor: a, [Actor: b,] (Goal: g | information: i | permission: p) {[, (**for** j **to** j **in** j **by**), (Actor: c | Goal: g | Time: t | Transmission Type: tt)] | BelievabilityCheck Type: bt)]}**)**

The syntax of the language can be described as follows, **bold** is a language keyword, definitions are represented as =, alternations are enclosed in round brackets () separated by |, optional elements are enclosed within square brackets [], and repetition is enclosed with braces {}. In what follows, we discuss the three types of IQ policies:

Permit policy is used to define the activities that an actor is allowed to perform over information, and it can be represented in BNF notation as follows:

Permit = **permitted_policy_name(**Actor: a, information: i [, (**for** | **to**), (Actor: c | Goal: g)]**)**

For example, the following permit policies are used to represent a stock investor that is permitted to produce its orders} (1), a stock trader that is permitted to read (2) and modify (3) Investor's_orders} for its goal Make profit by producing the right orders, and it is also permitted to send Investor's_orders to a stock market (NASDAQ) (4).

(1) **permitted_produce(**Investor, Investor's_orders**)**
(2) **permitted_read(**Trader, Investor's_orders, **for**, Make_profit_producing_orders**)**
(3) **permitted_modify(**Trader, Investor's_orders, **for**, Make_profit_producing_orders**)**
(4) **permitted_send(**Trader, Investor's _orders, **to**, NSQ**)**

Forbid policy is used to define the activities that an actor is prohibited to perform over information, and it can be represented in BNF notation as follows:

Forbid = **forbidden_policy_name(**Actor: a, information: i [, (**for** | **to**), (Actor: b | Goal: g)]**)**

For example, the following forbid policies can be used to represent a trader that is forbidden to produce Investor's_orders (e.g., does not have the required permissions) (1), and it is forbidden to read and modify (2-3) Investor's_orders by its goal *analyzing the market*. Finally, the trader is forbidden to send Investor's_orders to *NYSE* (4).

(1) **forbidden_produce(**Trader, Investor's_orders**)**
(2) **forbidden_read(**Trader, Investor's_orders, **for**, analyzing_the_market**)**
(3) **forbidden_modify(**Trader, Investor's_orders, **for**, analyzing_the_market**)**
(4) **forbidden_send(**Trader, Investor's_orders, **to**, NYSE**)**

Obligate policy is used to specify the activities that actors must perform over information, and it can be represented in BNF notation as follows:

obligate = **obligated_policy_name(**Actor: a, [Actor: b,] (Goal: g | information: i | permission: p) {[, (**for** | **to** | **in** | **by**), (Actor: c | Goal: g | Time: t | Transmission Type: tt)] | BelievabilityCheck Type: bt)]}**)**

Obligate policies cover five different types of activities, namely: produce, read, provide, send and monitor. Each of them is described as follows:

(1) Obligate *produce.* in such policy an actor (e.g., *CTA*) is obligated to apply a believability check mechanism while producing information (e.g., CTS-info), which help in avoid producing unintended information. For example, the following policy means that NYSE is obligated to apply a Believability Check while producing a CME_CB_info by its goal *analyze trading environment*:

obligated_produce(NYSE, NYSE_CB_info, **for**, analyze_trading_environment, **by**, BelievabilityCheck**)**

(2) Obligate *read.* we have defined two types of obligate read policies:
(i). an actor is obligated to apply a believability check mechanism while reading information, which helps in avoid depending on unbelievable information. For example, the following policy means that NYSE is obligated to apply a Believability Check while reading a NYSE_CB_info by its goal *manage trading environment*:

obligated_read(NYSE, NYSE_CB_info, **for**, manage_trading_environment, **by**, BelievabilityCheck**)**

(ii). an actor is obligated to read information within a predefined period of time delay, which helps in avoid information inconsistency within the system. For example, the following policy means that NYSE is obligated to read CME_CB_info by its goal *manage trading environment* within 0 time delay:

obligated_read(NYSE, CME_CB_info, **for**, manage_trading_environment, **in**, 0**)**

(3) Obligate *provide.* we differentiate between two types of provide obligate policies:
(i). an actor is obligated to provide information to another one within a predefined period of time. For example, the following policy means that Stock trader is obligated to provide Trading_suggestions to Stock investor within 7 second.

obligated_provide(Trader, Invester, Trading_suggestions, **in**, 7**)**

(ii). an actor is obligated to provide information to another one through a specific provision mean (e.g., Integrity Preserving (IP) provision). For example, the following policy means that investor is obligated to provide *its orders* to the trader through IP Provision:

obligated_provide(Investor, Trader, Investor's_orders, **by**, IP_Provision**)**

(4) Obligate *send.* in such obligate policy an actor is obligated to send information to its predefined destination within a predefined period of time. For example, the following policy means that the Trader is obligated to send Invester's_order to the NASDAQ in 10 seconds:

obligated_send(Trader, Invester's_order, **to**, NSQ, **in**, 10)

(5) Obligate *monitor.* in such obligate policy an actor is obligated to monitor another actor for producing information or monitor a delegated goal/permission. For example, the following policy means that NASDAQ is obligated to monitor the trader for the orders it produces:

obligated_monitor(NSQ, Trader, Trader's_orders**)**

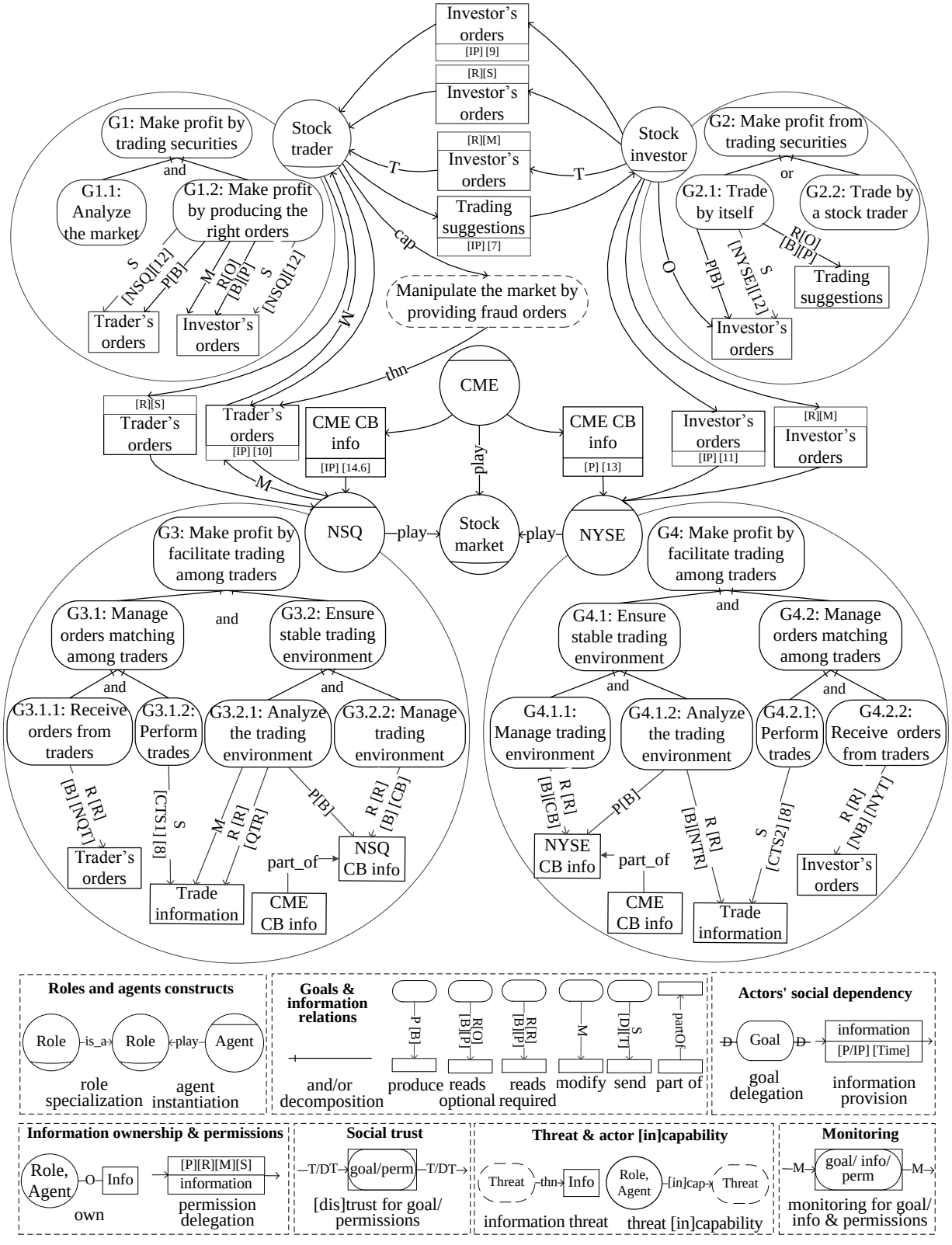

Figure 1 A partial goal model concerning the stock market system

Rules for the Automated Derivation of IQ Policy Specifications

In what follows, we discuss how the final IQ specifications can be derived from the requirements model. In particular, we define three sets of derivation rules (shown in Table 1), namely: permit, forbid and obligate policy derivation rules that are used for the automated derivation of IQ specifications from the requirements model in terms of IQ policy specification language; in what follows we discuss each of these sets:

Permit policy derivation rules: are used to derive allowed actors' activities concerning information, and represent them in IQ specification language (***permit policies***). In particular, rules **P1-5** are used to identify the permitted actors' activities over information based on the permissions they have taking into consideration that actors that might be prevented from performing some activities (e.g., conflict of interests).

P1: states that an actor is permitted to produce an information item, if it has the produce permission, and it is not forbidden, by any reason, from producing such information.
P2: states that an actor is permitted to read an information item for any goal that is responsible of its achievement, if it is the owner of such information, and there is no reason forbidding it from reading such information.
P3: states that an actor is permitted to read an information item for a specific goal that is responsible of its achievement, if it has the related read permissions, and it is not forbidden, by any reason, from reading such information.
P4: states that an actor is permitted to send an information item to a specific actor, if it is responsible of a goal that sends such information to the actor, and it has the related send permissions.
P5: states that an actor is permitted to modify an information item by a specific goal, if it is responsible of the goal that modifies such information, and it has the related modify permissions.

For example, in Figure 1 the *stock investor* is permitted to produce, read, modify and send its own information *investor's orders* by any goal that *the investor* is responsible of, and it is also permitted to delegate such permissions to any other actor. While *the trader* is permitted to read and modify *investor's orders* only by its goal ``Make profit by trading securities", and it is permitted to send *investor's orders* only by the same goal and only to *NASDAQ (NSQ)*.

Forbid policy derivation rules: are used to derive prohibited actors' activities concerning information, and represent them in IQ specification language (***forbid policies***). In particular, rules **F1-5** are used to identify the activities that an actor is forbidden to perform over information.

F1: states that an actor is forbidden to produce an information item, if it plays conflicting roles concerning the production of such information.
F2: states that an actor is forbidden to read an information item, if it plays conflicting roles concerning the read of such information.
F3: states that an actor is forbidden to read an information item for any goal that is responsible of its achievement except the goal(s) that reads such information, and the actor has been granted the read permissions to achieve it/them. This rule is used to prevent actors from using (reading) information for any goal beside the one they have been granted the read permissions for.
F4: states that an actor is forbidden to modify an information item by any goal that is responsible of its achievement except the goal(s) that modifies such information, and the actor has been granted the modify permissions to achieve it/them.
F5: states that an actor is forbidden to send an information item to any actor, except the actor(s) that has been granted the send permissions to send information to it/them.

For example in Figure 1, *the trader* is forbidden to read *investor's orders* by its goal ``analyze the market", and it is forbidden to send *investor's orders* by any goal except ``Make profit by producing the right orders" (e.g., ``analyze the market") to any actor but *NASDAQ (NSQ)*.

Table 1. Rules for the Automated Derivation of IQ Specifications

Permit policies
P1 permitted_produce(A,I) **:-** has perm(p,A,I), not forbidden produce(A,I).
P2 permitted_read(A,I,for,G) **:-** own(A,I), is responsible(A,G), not forbidden read(A,I,for,G).
P3 permitted_read(A,I,for,G) **:-** has perm(r,A,I), is responsible(A,G), read(POU,G,I),not forbidden read(A,I,for,G).
P4 permitted_send(A,I,to,B) **:-** has perm(s,A,I), is responsible(A,G), send(T,G,B,I).
P5 permitted_modify(A,I,for,G) **:-** has perm(m,A,I), is responsible(A,G), modify(G,I).
Forbid policies
F1 forbidden_produce(A,I) **:-** play(A,R1), play(A,R2), conflict roles produce(R1,R2,produce,I).
F2 forbidden_read(A,I,for,G) **:-** play(A,R1), play(A,R2), is responsible(A,G), read(POU,G,I), conflict roles read(R1,R2,read,I).
F3 forbidden_read(A,I,for,G1) **:-** has perm(r,A,I), is responsible(A,G), read(POU1,G,I), is responsible(A,G1), not read(POU2,G1,I), G != G1.
F4 forbidden_modify(A,I,for,G1) **:-** has perm(m,A,I), is responsible(A,G), modify(G,I), is responsible(A,G1), not modify(G1,I) , G! = G1.
F5 forbidden_send(A,I,to,C) **:-** has perm(s,A,I), sender(T1,A,B,I), actor(C), not sender(T2,A,C,I), #int(T1), #int(T2), B!=C.
Obligate policies
O1 obligated_produce(A,I,by, bck) **:-** is responsible(A, G), produce(bck, G, I, T).
O2 obligated_read(A,B, by, bck) **:-** is responsible(A, G), read(RO, POU, bck, G, I)
O3 obligated_read(A,I,in,0) **:-** interdependent readers(A,B,I).
O4 obligated_provide(A,B,I,in,T) **:-** provideChain(,T,A,B,I).
O5 obligated_provide(A,B,I,by,ip) **:-** provideChain(ip,T,A,B,I).
O6 obligated_send(A,I,to,B,in,T) **:-** sender(T,A,B,I).
O7 obligated_monitor(A,B, (G|I|P)) **:-** monitorChain(A, B, (G|I|P)).

Obligate policy derivation rules: are used to derive obligated actors' activities toward information, and represent them in IQ specification language (***obligate policies***). In particular, rules **O1-7** are used to identify the activities that an actor is obligated to perform over information.

O1: states that an actor is obligated to apply a believability mechanism check while producing information.
O2: states that an actor is obligated to apply a believability mechanism check while reading information.
O3: states that all ***interdependent readers*** for information are obligated to read such information with no delay (`0').
O4: states that an actor is obligated to another one to provide information within the provision time it claims.
O5: states that an actor is obligated to another one to provide information through an integrity preserving transmission.
O6: states that an actor is obligated to send information to its intended destination within the send time it claims.
O7: states that an actor is obligated to another one to monitor the achievement of a goal/ producing an information item.

For example, in Figure 1 *the trader* is obligated to apply a believability check while producing (**O1**) and reading (**O2**) *Trader's orders* and *investor's orders* respectively. While both *NYSE* and *NASDAQ* are obligated to read ``CME CB info" with no delay (`0'), since they are *interdependent readers* for such information (**O3**). At the other hand, *the trader* is obligated to provide ``Trading suggestions" information to the investor within 7 seconds (**O4**), since it is the provision time among both them. In addition, *the trader* is obligated to provide ``Trading suggestions" information to the investor through IP-provision (**O5**). While *the trader* is obligated to *the investor* to send its information (*investor's orders*) to *NASDAQ* within 12 second as it claims (**O6**). Finally, NASDAQ is obligated to monitor *Trader's orders* (**O7**).

Bibliography

[1] Daniel D McCracken and Edwin D Reilly, "Backus-naur form (bnf)," *Encyclopedia of Computer Science*, pp. 129–-131, 2003.