**PAGE EDIT FROM ORIGINAL PROPOSAL & NEW TABLE + NEW FIGURE**

* 20 Must fix the red part to better differentiate related work and mine
* 21 + 27, new paper in related work
* P36, reference to new Table, X1 (the simplified goal condition in Token)
* From P39, remove some part before **Token** and explain TOKEN system.

**THING LEFT NOT FINISH !!!!**

**-[-1] QuestGiver current is just simply >>>> select random char/item as quest giver and get it’s questLevel. Then when selecting token, don’t pick any object with higher level**

**No need to report to quest giver???? (reason =, some game just finish quest when complete, no need to report back?)**

**^ Ask AJ if ok to reason with this and left it as is. ALSO do need to explain?**

-0. Check that the rule on creating token on what Component node are correct.

-1 Token to every subquest node, unless that subquest is brokendown into NULL.

-Any node with token that are brokendown only into NULL, are deleted.

1. Make proper way to select NPC or location for INFO token

From QuestGeneratorUtility !!!!

// 11

// Info >>> Info of Item = Select Random NPC? and condition = [player at same NPC location]?

// 12

// Info >>> Info of Character = Select Random NPC? and condition = [player at same NPC location]?

// 13

// Info >>> Info of Location = Select Random NPC? and condition = [player at same NPC location]?

//

// 21

// Loc >>> Loc of Item = Get the Location of the Item, and condition = [player at that location]?

// 22

// Loc >>> Loc of Character = Get the Location of the Character, and condition = [player at that location]?

// 23

// Loc >>> Loc of Location = just [player same location]?

//

// 31

// itself >>> itself of Item = Depend on Component type (dmg, give, etc.)

// 32

// itself >>> itself of Character = Depend on Component type (capture, free, etc.)

// 33

// itself >>> itself of Location = Depend on Component type?????

//

//

// IF(contain '4') == null and ignore >> go to next one.

The 11, 12, 13 now use ‘sameLocation’ instead of ‘currentLocation’ as of 7-1-2019

This is only apply to NPC , (do not assign ‘sameLocation’ to item) <<< (it may create weird bug)

OTHER ‘currentLocation’ that is replaced with ‘sameLocation’

[Below = only if Object = NPC]

case "escort" :

case "follow" :

[Below = only if Object = NPC]

case "examine":

case "explore":

case "listen":

case "goto":

f

1. Separate the Location goal with same-NPC-Location goal  **\*\*\* DONE \*\*\***

This is done by making new gameCondtion call [sameLocation], which came in this format

[Char\_name, sameLocation, Char2\_name,z,zz,zzz]

-This condition is mixed with other GC at start

-However, when the Prolog is query, it will start by separating these ‘special’ condition (which can’t be identify if satisfy or not by using just ‘member(AC,GC)’ )

-After that we will have ‘GC’ (normal goal condition) and ‘GL’ (special goal condition that require more complex process to identify if they are satisfied yet).

1. Make some real Stealth Simulation

// VERY HARD TO SIMULATE, let just use this as temp

**case** "spy":

**case** "stealth":

curChar = (Character) componentinput.getComponentObject();

output = "1:";

output += curChar.getName();

output += ":listStatus:" + "got\_sneak";

outputList.add(output);

**return** outputList;

1. Make a functioning ‘getCharacterName’

**Major Update NOTE:**

8-1-2019:

-Implement [checkSpecialCondition(GL,AC)] to Prolog (to check for ‘sameLocation’)

**-GL will now contain all ‘special’ condition goal (such as ‘sameLocation’) that require complex checking procedure (anything more complex than [member(Goal,GameState)] )**

-AC and AL will now be merged to just AC (AL is too redundant)

-As consequence AL will be deleted from ALL **(HAS BACKUP in Prolog/8-1-2019)**

**Some Simplification that happen in the work**

1. All money related action are not really ‘simulated’. For example, [hire to attack] or [exchange] don’t really give the NPC / Character any actual ‘money’ or ‘item’ for their service.

**This is because the aspect of basic ‘money’ is assumed to be possible from start to avoid unnecessary coding of ‘farming / grinding’ aspect of the simulation. Which is out of scope of this work.**

30-1-2019

* Change the **append( [char\_die], P, P2)** to **append( P, [char\_die], P2).**

8-2-2019

* **QueryAll** refuse to work, >>>> fixing by just using **QueryOnce**, but alter the Prolog code to print all result path by using [force fail] in [write\_to\_file].
* This will make the **QueryOnce**(query) return ‘null’ string in Java, but all the path will still be written in the folder as file.z

12-2-2019

* Finally, everything run as intended
* However, super un-optimized, finding path too so long
* Save the ‘working’ folder in USB, in Backup >>> name [Prolog unoptimized working]
* Starting to work on Optimization
* Fixing duplicated Path by another makeshift fix  
  **Fixing by Implementing “pathExist(0)”**

**This fix by everytime a path is found, it will added using assert**

**Then when checking if quest objective is reached, it will check if the current path already exist in ‘assert/1’ yet.**

**pathExist**(**0**).

**writeToFile**(**GC**,**GL**,**AC**,**AR**,**LA**,**P**,**LC**,**PF**)

:-

**assert**(**pathExist**(**P**)),

**Change from Proposal NOTE:**

* **RED** = Information had been simplified, just by holding the related item or being at the same location is enough. There is no real ‘information’ object being simulated within the program.
* As subsequence, Talking / Listen / Report are also simplified to just having the character at same place as player to satisfy their condition.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Component** | **Start State** | **Goal State** |
| **1.** | ɛ | None. | None. |
| **2.** | Capture **X** | **Somebody** is there. | **They** are your prisoner. |
| **3.** | Damage **X** | **Somebody** or **something** is there. | **It** is more damaged. |
| **4.** | Defend **X** | **Somebody** or **something** is there. | Attempts to damage **it** have failed. |
| **5.** | Escort **X** | **Somebody** is there. | **They** will now accompany you. |
| **6.** | Examine **X, Y** | **Somebody (Y)** or **something (Y)** is there. | You have **information (X)** about it. |
| **7.** | Exchange **X|Y, Z** | **Somebody** **(Z)** is there, they have **something** **(X)** and you have **something** **(Y).** | You have **(Y)** and they have **(X).** |
| **8.** | Experiment **X, Y** | **Something (Y)** is there. | Perhaps you have learned **information (X)** what **it (Y)** is for. |
| **9.** | Explore **X** | **There** exist. | Wander around **there** at random. |
| **10.** | Follow **X** | **Somebody** or **something** is there. | You will now accompany **them**. |
| **11.** | Free **X** | **Somebody** is there and is prisoner. | **They** are no longer prisoner. |
| **12.** | Gather **X** | **Something** is there. | You have **it.** |
| **13.** | Give **X, Y** | **Somebody** **(Y)** is there, you have something **(X).** | They have **it (X)**. |
| **14.** | Goto **X** | You know **where** to go and how to get **there**. | You are **there**. |
| **15.** | Kill **X** | **Somebody** is there. | **They** are dead. |
| **16.** | Listen **X, Y** | **Somebody (Y)** is there. | You have some of their **information (X).** |
| **17.** | Read **X, Y** | **Something (Y)** is there. | You have **information (X)** from it. |
| **18.** | Repair **X** | **Something** is there. | **It** is fixed, built or resolved. |
| **19.** | Report **X, Y** | **Somebody (Y)** is there. | **They (Y)** have **information (X)** that you have. |
| **20.** | Spy **X, Y** | **Somebody (Y)** or **something (Y)** is there. | You have **information (X)** from **it (Y)**. |
| **21.** | Stealth **X** | **Somebody** is there. | Sneak up on **them**. |
| **22.** | Take **X, Y** | **Somebody** **(Y)** is there, they have **something** **(X).** | You have **it (X)** and they don’t. |
| **23.** | Use **X** | **Somebody** or **something** is there. | **It** has affected characters or environment. |
| **24** | wait | None. | Wait for something to happen. |

**Thing to remember: Data Structure policy**

1. -The path from Prolog will be written as below (before reverse)

-When read into Java, the whole list have to be reversed in order to get the proper order of action performed

%[player,jack,direct\_attack,quest\_start] - before reverse

%[quest\_start,direct\_attack,jack,player] - after reverse

* 1. ABOVE code mean that all ‘append’ must follow the following structure

append(**[Thing\_added]**,AC,AC2)

* 1. Now let just add to right side so no need to reverse.

append(AC, **[Thing\_added]**, AC2)

1. –The [action] (direct\_attack) as shown above, will be followed by objects (the acted upon, and the actor). The number of objects is depended on the action (the java must use **case’ACTION’** to read them properly)
   1. **Direct\_attack, jack, player** 
      1. jack (the one being attack)
      2. player (the attacker)
   2. **exchange, coin,player, potion, merchant1**
      1. coin = (what player give, itemID)
      2. player = (owner of 1st item)
      3. potion = (what merchant1 give, itemID)
      4. merchant1 = (owner of 2nd item)
   3. **pick\_up, itemID, PICKER, SOURCE**
2. – The (hire\_to\_attack,player,AtName,TarName) structure is that the [action] is always follow by [ACTOR] then [TARGET] if possible.
3. –For generic gameCondition
   1. [NAME: gamecondition1 : value1 : gamecondition2 : value2 : itemID (if exist) ]
   2. [player:currentLocation:market:z:zz:zzz] =
   3. [player,currentLocation,market,z,zz,zzz] (Prolog)
4. –For relationship GameCondition
   1. [relationshipTYPE: actor1 : actor2]
   2. [friend: player: jack]
   3. [friend, player, jack]
5. The p (path) will have it’s 1st slot be used to tell the Prolog what’s the file name and file path.

Ex. If a file is the 2nd component, it will have ‘b’ coding. If this file came from situation ‘a1’, the whole file will be name **[a1b1.txt].** The a1b is passed from Java, and the last [**1**] is generated by Prolog (see paint jpeg as super basic idea).

The last number will be responsible by the Prolog. The Java will be responsible for the ‘a1’ part.

When sending this p(path) information to Prolog, the java will send p(path) as **[a1b]** in the query.

7.1 Change from 7

Now, P will have 2 slot at start, which look like [a1, b] OR [#,a] (1st folder)

The last (b) will be used for folder creation

The 2nd Last (a1), will be used for file Creation

Java read these info by….

The [a1] came from file name (from the current accessing folder)

The [b] came from the (.tostring()) of current number of Component

LIST OF ITEM

1XXXYY (consumable, natural)

-100101,"berry",

-100201,"poison\_plant"

2XXXYY (consumable / crafted)

-200101,"potion\_poison"

-200201,"potion\_heal"

-200301,"antidote"

3XXXYY (tool)

-300101,"lockpick"

5XXXYY (Equipment)

-500101,"dagger"

9XXXYY (Quest item / Unique item)

-900101,"diamond"

-900201, “treasure map”

-coin (for bribe, buy item)

LIST OF PATH NAME

[friend\_heal\_poison] + healed + healer

[criminal\_resist\_soldier\_capture\_and\_die] + NPC who resist & die

[criminal\_escort\_to\_jail] + criminal + the soilder

[poison] + Target + poisoner

[goto] + From + Destination + actor

LIST OF SKILL

**Misc Note:**

Why Prolog…..

- Mainly => no eclipse plugin for any other logic languages that seem to fit better to my work than prolog.

-

Relationship

-Not written at 1st cause there’s no [Component] that require ‘relationship’ to be achieved in GoalState to advance quest forward

Exchange

-Wasn’t really written as true ‘exchange’ in GoalState because too complex.

-Thus written as current for ‘simplify’

// Each object (character, item, etc.) should have levelQuest to determine what is 'acceptable' as quest

// objective depended on quest giver (farmer shouldn't give quest to kill dragon)

int levelQuest;

IN GAMESTATE & GAMEWORLD

//COMMENT 6-12-2018 This is comment out since All Item are now in either Character or in Location

/\*

public ArrayList<GameCondition> getListGameConditionItem(){

return listDesireItem;

}

\*/

THERE’s no item consumption, used item don’t disappear

-Used item will still be in the inventory, this is because deleting the item may cause poor interaction

EX: if can craft item, may loop making item, then consume it then make again

THERE’s no

**BIG DIVERGE FROM SOURCE PAPER**

- Let select a thing 1st as quest giver, then select token as normal, however token can’t have higher levelQuest than the quest giver

Quest giver always has ‘report’ at the end of quest?

PROBLEM: quest giver may not interact well with token system >>

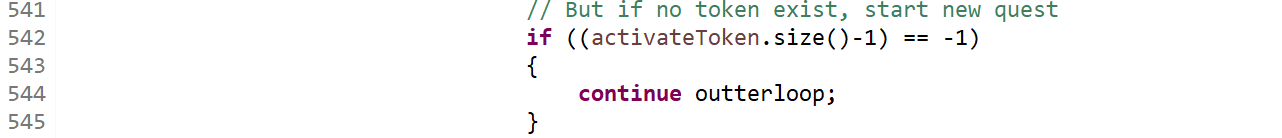
- (quest giver may be killed mid quest) >> **solve by just ignoring this, quest starter doesn’t necessary have to be alive when quest end, some story quest may start with king, then MC uncover his wrong doing and kill the king, then report quest to someone else.**

- (quest giver may not exist in quest at all; as in quest don’t interact with quest giver such as getting berry to NPC\_A, NPC\_A has high chance to not be quest giver)

**BUG THAT FIX WITH MAKESHIEFT hotcode**



This part



Is created because the [activateToken.get(activateToken.size()-1] produce -1 value

**(AKA: The activateToken list is empty).**