Article

Robots arm motion representation in Petri NETS using sequent calculus

Syed Uzair Ahmad
Hazara University, Manshera, Pakistan
E-mail: ssuab.kk@gmail.com

Received 1 August 2015; *Accepted 5 September 2015*; *Published online 1 December 2015*

Abstract
There are many sort of motion in robots structure. Such as the robot locomotion robot jumps robots picking and so on but all are presented through Petri NETS. The one motion which is also the important one and most worthy motion of the robots is the robots arm motion. Which till yet not represented through Petri NETS. In this paper we are going to represent the motion of the robot arm in different angles and different aspect, such as up, down, circular, back and front moment of the robot arm, through Petri net we can present the complex form of motions into simplex paths.

Keywords Petri Nets; 2D; 3D; sequent calculus.

Selforganizology
ISSN 2410-0080
URL: http://www.iaees.org/publications/journals/selforganizology/online-version.asp
RSS: http://www.iaees.org/publications/journals/selforganizology/rss.xml
E-mail: selforganizology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

Firstly it is stated that all that work which we represent in graph become much easy to understand this paper also consist of such like related work the [1.0] describes the motions of different types farther it defines the different aspects of 2D and 3D and that motion are compared with human motion. Farther in detail the part [2.0] goes to the definition and some important position and transition of the Petri NETS the importance as described that a Petri net is a transition graph (also known as a place/transition net or P/T net). The Petri Nets Applications is also described in part [3.0]. The next forward step in [4.0] we are going to show the implementation on the robot arm motion in Petri NETS. This portion consist some of the equations through easily fallow the rule of motion. The [4.0] consists farther division as in [4.1] we present the motion Up and Down in Petri NETS notation as that motion was simply described in [4.0]. The mathematical proof is defined in the part [4.2]. Definition of Sequent Calculus is described and symbols is shown respectively [4.3] [4.4]. Proof of the above motion done in the section [4.5].the circular motion is presented the part consists of two types of motions Right Hand side and Left Hand side Circular motion [4.6] and its proof is given the section [4.7].and the impacts of this paper on RE is described in [5.0].

1.0 Robot motions

The motion of humanoid robot arm is important for the communication with the people (Fig. 1). In this paper the motion of the robot is represent in the form of Petri NETS. According to Kim et al. (2006), the robot arm motion is presented mathematically by using tool, Response Surface Method (RSM). The proposed method was evaluated to generate human like arm motion when the robot was asked to move its arms from a point to another point including the rotation of hand.

Fig. 1 The definition of elbow elevation angle for a human arm.

May the robot motion be in 2D or in 3D, Lumelsky (1991) present a general motion planning in 2D and 3D environment. Each joint of the robot arm manipulator can be either revolute or sliding. All different sort of motion is present with different views and angle of robot arm and the linkages of the robots arm in different prospective (Fig. 2).

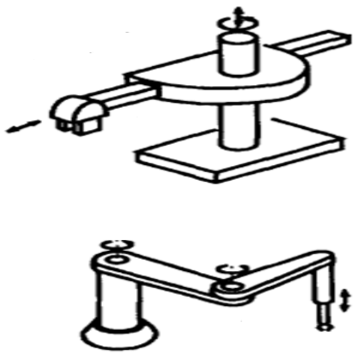


Fig. 2 Basic robot arm linkages.

A technique to renew a human-like arm motion by modifying and scaling a human arm motion is obtainable. The humanoid robot may not obtain all the necessary human-like motions from motion detain data.

When the robot communicates with a person, the robot has to keep notice to the person by aligning the direction of the motion. The robot needs to modify human arm motions and regenerate new human-like arm motions without losing the original meanings. The motion of drawing multiple circles with a various radius and direction is examined (Kim et al., 2007).

2.0 The Petri Nets with types of motion

A Petri net (also known as a place/transition net or P/T net) is one of several mathematical modeling languages for the explanation of distributed systems. A Petri net is a directed bipartite graph, in which the nodes represent transitions (i.e. events that may occur, signified by bars) and places (i.e. conditions, signified by circles). The directed arcs describe which places are pre- and/or post conditions for which transitions (signified by arrows) occurs. Some sources state that Petri NETS were invented in August 1939 by Carl Adam Petri.

The motion of Petri net is in the form of loop and simple motion form one point to another is presented in the given model (Kim et al., 2007).

Fig. 3 presents "graphically structural and marking conditions of a kit of four particular cases of reduction rules. It is not difficult to observe that they preserve such properties as livens, the bounds of places and, if the second place has only one input transition, reversibility" (DiCesare et al., 1993).

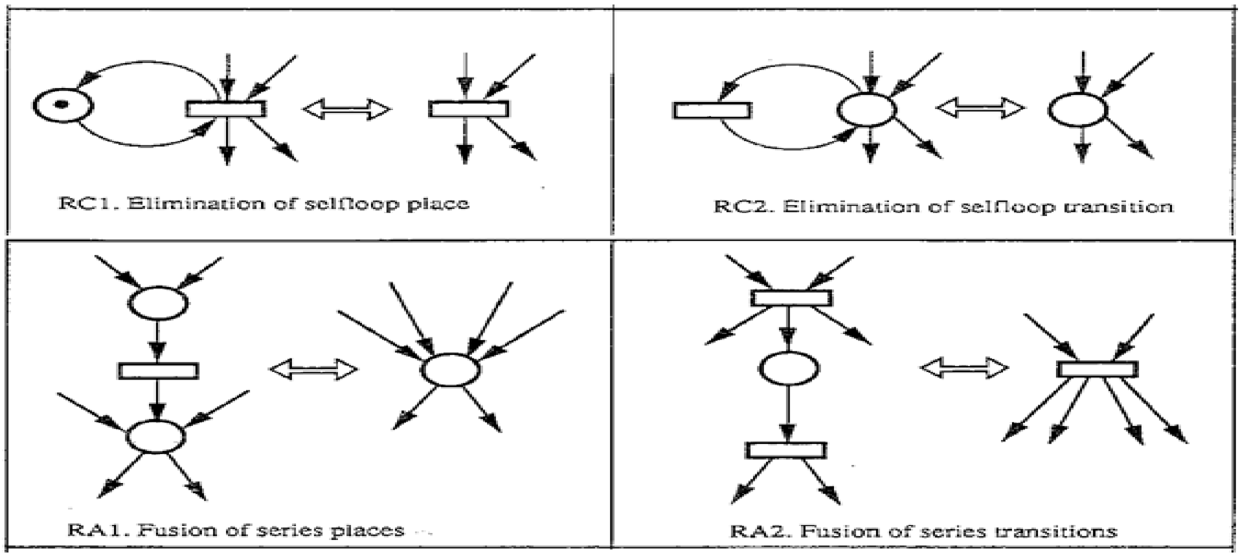


Fig. 3 Petri Nets and types of motion.

3.0 Petri Nets applications

Petri Nets are graphical and mathematical tool used in many different science domains. Their characteristic features are the intuitive graphical modeling language and advanced formal analysis method. The concurrence of performed actions is the natural phenomenon due to which Petri Nets are perceived as mathematical tool for modeling concurrent systems. The nets whose model was extended with the time model can be applied in modeling real-time systems. Petri Nets were introduced in the doctoral dissertation by K.A. Petri, titled "Kommunikationmit Automaten" and published in 1962 by University of Bonn. During more than 40 years of development of this theory, many different classes were formed and the scope of applications was extended.

Depending on particular needs, the net definition was changed and adjusted to the considered problem. The unusual "flexibility" of this theory makes it possible to introduce all these modifications. Owing to varied currently known net classes, it is relatively easy to find a proper class for the specific application. The present monograph shows the whole spectrum of Petri Nets applications, from classic applications (to which the theory is specially dedicated) like computer science and control systems, through fault diagnosis, manufacturing, power systems, traffic systems, transport and down to Web applications. At the same time, the publication describes the diversity of investigations performed with use of Petri Nets in science centers all over the world.

4.0 Implementation of Petri Net in Robot Arm Motion (My Potential Contribution)

According to the physics the turning effect of the body is called torque now by the formula we can represent the motion as (Fig. 4)

$$T=F*D$$

where F is force and D is the distance from the start point.

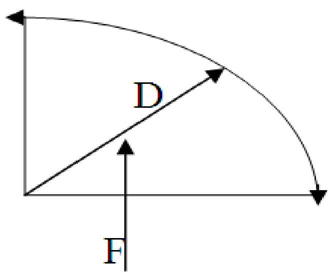


Fig. 4 Torque

The force will need according to the distance from the start point distance increase the force will decrease and conversely.

Now if we apply this procedure on the arm of human, then circular and angular motions takes place as Fig. 5 and 6.

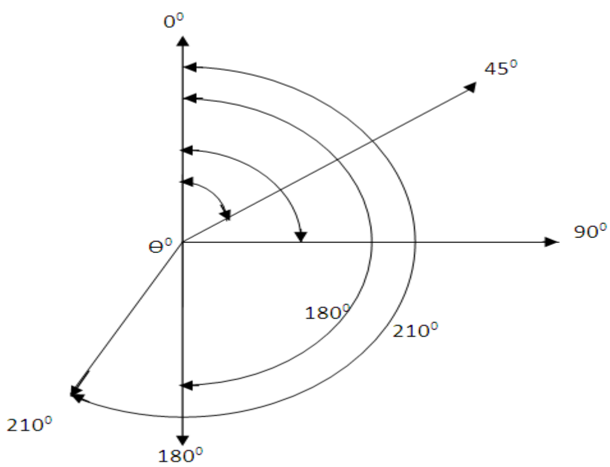


Fig. 5 Angular representation of human arm motion

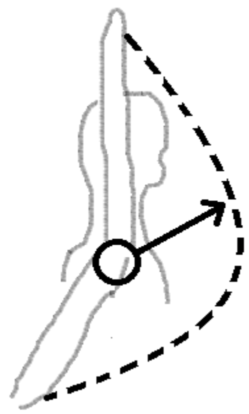


Fig. 6 Human arm motion.

Now the circular motion is that motion if a body moves in a circular path with a uniform speed, it is said to be in uniform motion. According to the Newton second law

$$F=ma$$

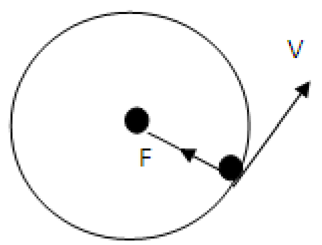
The acceleration produced by centripetal force

$$a=v^2/r$$

So we can write

$$F=mv^2/r$$

where v is the uniform speed, F is the force, m is the moving in a circle, and r is the radius.



Now to present the motion of arm in Petri NETS according to the above equation and rule we can get a result as

4.1 Up and down motion of arm in Petri NETS (Fig. 7)

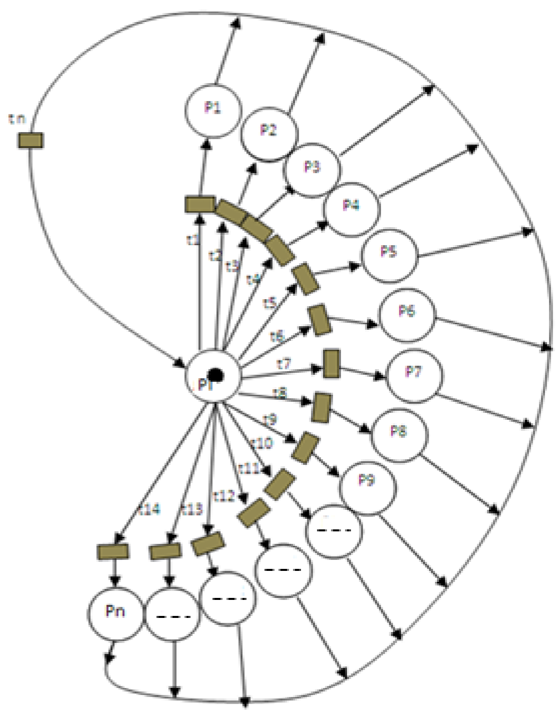


Fig. 7 Up and down motion of arm in Petri NETS.

4.2 Explanation

According to the formula we can explain as that when a token is fire from place pi and takes the direction of p1 and t1 through transition tn, it continue its loop until the place pi shift its token to p2 and t2 on the way it continue its loop until to the shifting of token to the another transition and so on the process is continue and the Petri net goes up and down word motion.

4.3 Definition of sequent calculus

Logical formulas are expressions constructed from predicate symbols and variables as their arguments with using connectives & (and), ∨ (or), ⊃ (implication), ¬ (negation), and quantifiers ∀ (universal), ∃ (existential). Yet another connective ~ (equivalence) is defined via others as A ~ B = (A ⊃ B) & (B ⊃ A). Optionally, terms constructed with using object functions and constants may be allowed as arguments of predicate symbols.

Sequents are expressions of the form Γ |- Δ, where Γ and Δ are (possibly empty) sequences of logical formulas. Γ is called the antecedent and Δ is called the consequent. A formal theory in which axioms and inference rulesare formulated in terms of sequents is called sequent calculus. In fact, there are several sequent calculi; these variants will be discussed later in section Formulations. The informal understanding of sequents is that the sequent $A_1, ..., A_m$|- $B_1, ..., B_n$ corresponds to the formula: A_1& ... &A_m|- B_1∨ ... ∨B_n

The only axiom schema of sequent calculus is A |- A, where A is any logical formula. There are two sorts of inference rules in sequent calculus: structural and logical. Every logical rule corresponds to a connective or quantifier. There are at least two logical rules for every propositional connective and every quantifier. One of them applies to the antecedent, whereas the other applies to the consequent. Further on, Greek letters denote sequences of formulas, and English letters denote individual logical formulas.

4.4 Symbols

Symbol	Name Should be read as Category	Explanation	Examples	Unicode Value	HTML Entity	LaTeX symbol
⇒ → ⊃	material implication implies; if .. then propositional logic,Heyting algebra	*A* ⇒ *B* is true just in the case that either *A* is false or *B* is true, or both. → may mean the same as ⇒ (the symbol may also indicate the domain and codomain of a function; see table of mathematical symbols). ⊃ may mean the same as ⇒ (the symbol may also mean superset).	$x = 2 \Rightarrow x^2 = 4$ is true, but $x^2 = 4 \Rightarrow x = 2$ is in general false (since *x* could be −2).	U+21D2 U+2192 U+2283	⇒ → ⊃	⇒ \Rightarrow → \to ⊃ \supset ⟹ \implies
⇔	material	*A*⇔*B* is true just in case either both *A*	$x+5=y+2 \Leftrightarrow x+3=y$	U+21D4	⇔	⇔

≡ ↔	equivalence if and only if; iff; means the same as propositional logic	and *B*are false, or both *A* and *B* are true.		**U+2261** **U+2194**	**≡** **↔**	**\Leftrightarrow** ≡**\equiv** ↔ **\leftrightarrow** ⟺**\iff**
¬ ~ !	negation not propositional logic	The statement ¬*A* is true if and only if *A* is false. A slash placed through another operator is the same as "¬" placed in front.	¬(¬*A*)⇔ *x*≠*y*⇔ ¬(*x*=*y*)	**U+00AC** **U+02DC**	**¬** **˜** ~	¬**\lnot or \neg** ∼**\sim**
∧ • &	logical conjunction and propositional logic	The statement *A* ∧ *B* is true if *A* and *B* are both true; else it is false.	$n < 4 \;\wedge\; n > 2 \;\Leftrightarrow\; n = 3$ when n is a natural number.	**U+2227** **U+0026**	**∧** **&**	∧**\wedge or \land** **\&**[1]
∨ + ∥	logical disjunction or propositional logic	The statement *A* ∨ *B* is true if *A* or *B* (or both) are true; if both are false, the statement is false.	$n \geq 4 \;\vee\; n \leq 2 \;\Leftrightarrow n \neq 3$ when n is a natural number.	**U+2228**	**∨**	∨**\lor or \vee**
	exclusive disjunction	The statement *A* ⊕ *B* is true when either A or B, but not both, are true.*A*	(¬*A*) ⊕ *A* is always true, *A* ⊕ *A* is always false.	**U+2295**	**⊕**	⊕**\oplus**

⊕ ⊻	xor propositional logic,Boolean algebra	⊻ *B* means the same.		**U+22BB**		⊻**\veebar**
⊤ T 1	Tautology top, verum propositional logic,Boolean algebra	The statement ⊤ is unconditionally true.	*A* ⇒⊤ is always true.	**U+22A4**	**T**	⊤**\top**
⊥ F 0	Contradiction bottom, falsum propositional logic,Boolean algebra	The statement ⊥ is unconditionally false.	⊥⇒ *A* is always true.	**U+22A5**	**⊥** **F**	⊥**\bot**
∀ ()	universal quantification for all; for any; for each first-order logic	∀ *x*: *P*(*x*) or (*x*) *P*(*x*) means *P*(*x*) is true for all *x*.	∀ $n \in \mathbf{N}$: $n^2 \geq n$.	**U+2200**	**∀**	∀**\forall**
∃	existential quantification there exists	∃ *x*: *P*(*x*) means there is at least one *x* such that *P*(*x*) is true.	∃ $n \in \mathbf{N}$: *n* is even.	**U+2203**	**∃**	∃**\exists**

	first-order logic					
∃!	uniqueness quantification there exists exactly one first-order logic	$\exists!\, x$: $P(x)$ means there is exactly one x such that $P(x)$ is true.	$\exists!\, n \in \mathbf{N}$: $n + 5 = 2n$.	**U+2203 U+0021**	**∃ !**	∃!**\exists !**
:= ≡ :⇔	definition is defined as everywhere	$x := y$ or $x \equiv y$ means x is defined to be another name for y (but note that ≡ can also mean other things, such as congruence). $P :\Leftrightarrow Q$ means P is defined to be logically equivalent to Q.	cosh $x := (1/2)(\exp x + \exp(-x))$ A XOR $B :\Leftrightarrow (A \vee B) \wedge \neg(A \wedge B)$	**U+2254 (U+003A U+003D) U+2261 U+003A U+229C**	**:= : ≡ ⇔**	≔**:=** ≡**\equiv** ⇔ **\Leftrightarrow**
( )	precedence grouping parentheses, brackets everywhere	Perform the operations inside the parentheses first.	$(8 \div 4) \div 2 = 2 \div 2 = 1$, but $8 \div (4 \div 2) = 8 \div 2 = 4$.	**U+0028 U+0029**	**( )**	( )**( )**
⊢	Turnstile provable propositional logic,first-order logic	$x \vdash y$ means y is provable from x (in some specified formal system).	$A \rightarrow B \vdash \neg B \rightarrow \neg A$	**U+22A2**	**⊢**	⊢**\vdash**
⊨	double turnstile	$x \models y$ means x semantically entails y	$A \rightarrow B \models \neg B \rightarrow \neg A$	**U+22A8**	**⊨**	⊨**\models**

	entails					
	propositional logic,first-order logic					

4.5 Proof of the above motion

$$\frac{}{Pi \vdash P_1t_1} \quad \frac{}{Pi \vdash P_2t_2} \quad \frac{}{Pi \vdash P_3t_3} \quad \frac{}{Pi \vdash P_4t_4} \quad \ldots\ldots\ldots \quad \frac{}{Pi \vdash P_nt_n}$$

$$\frac{}{Pi \vdash P_1t_1 \oplus P_2t_2 \oplus P_3t_3 \oplus P_4t_4 \oplus \ldots\ldots \oplus P_nt_n}\ R\oplus$$

$$\frac{}{P \vdash P} \quad \frac{}{t \vdash t} \quad \text{id Rule}$$

$$\frac{}{P,t \vdash p \otimes t}\ R\otimes$$

$$\frac{}{p,t \vdash (p \otimes t) \oplus (p \otimes t)}\ R\oplus$$

$$\frac{}{p,t \vdash (p \otimes t) \oplus (p \otimes t) \ldots\ldots\ldots}\ R\oplus$$

$$\frac{}{p \vdash (p \otimes t) \oplus (p \otimes t) \ldots\ldots\ldots}\ \text{Cut}$$

$$\frac{}{p\,i \vdash (p1 \otimes t1) \oplus (p2 \otimes t2) \oplus \ldots\ldots\ldots \oplus (pn \otimes tn)}\ \text{proved}$$

4.6 Now for circular motion (Fig. 8 and 9)

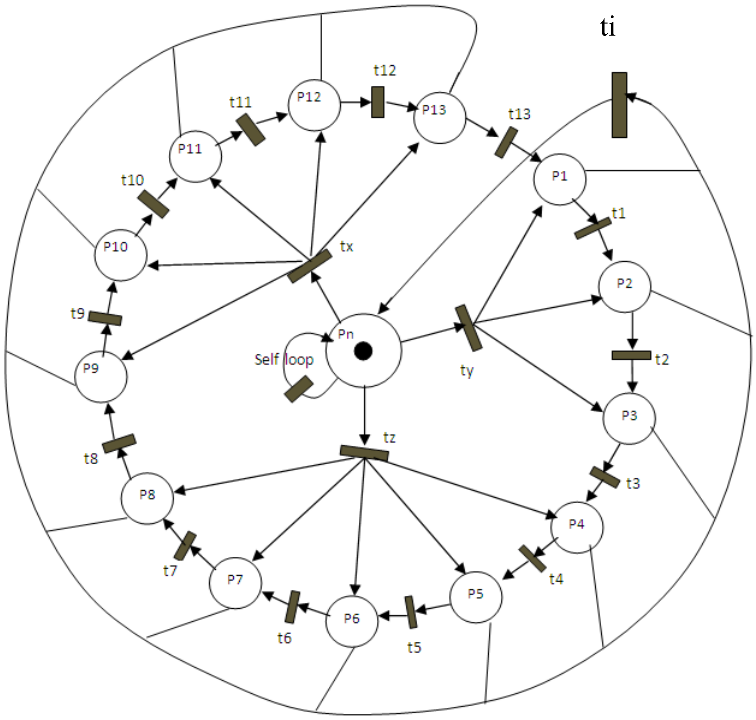


Fig. 8 Left side circular motion of arm in Petri NETS.

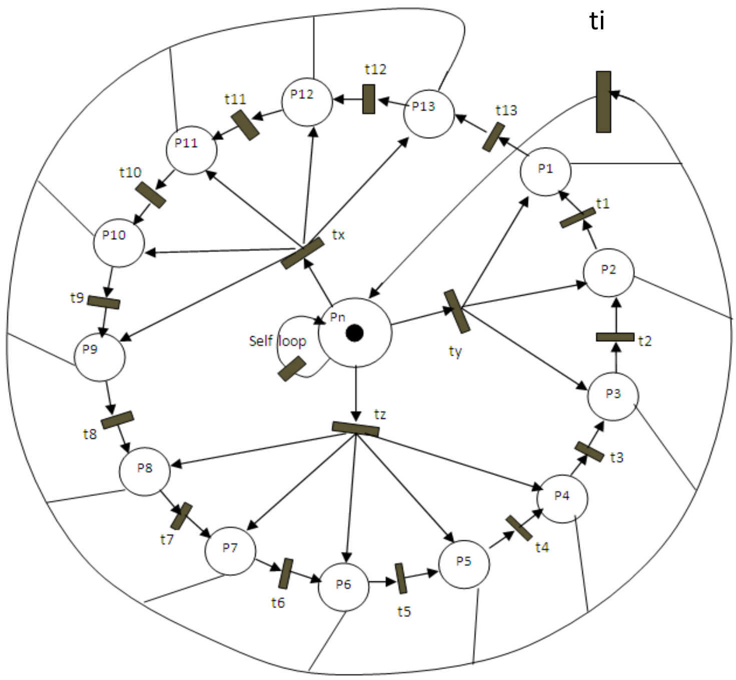


Fig. 9 Right side circular motion of arm in Petri NETS.

4.7 Proof

$$\frac{}{Pn \ \& \ Pn} = Pn \quad \text{id Rule}$$

$$\frac{}{Pn \vdash t_x \otimes !tn} \quad \frac{}{Pn \vdash t_y \otimes !tn} \quad \frac{}{Pn \vdash t_z \otimes !tn}$$

$$\frac{}{Pn \vdash (t_x \oplus t_y \oplus t_z) \otimes !tn} \ R\oplus, !c$$

Where

$$\frac{}{t_z \vdash P_4t_4 \otimes !tn} \quad \frac{}{t_x \vdash P_9t_9 \otimes !tn} \quad \frac{}{t_y \vdash P_1t_1 \otimes !tn}$$

$$\frac{}{t_z \vdash P_5t_5 \otimes !tn} \quad \frac{}{t_x \vdash P_{10}t_{10} \otimes !tn} \quad \frac{}{t_y \vdash P_2t_2 \otimes !tn}$$

$$\frac{}{t_z \vdash P_6t_6 \otimes !tn} \quad \frac{}{t_x \vdash P_{11}t_{11} \otimes !tn} \quad \frac{}{t_y \vdash P_3t_3 \otimes !tn}$$

$$\frac{}{t_z \vdash P_7t_7 \otimes !tn} \quad \frac{}{t_x \vdash P_{12}t_{12} \otimes !tn}$$

$$\frac{}{t_z \vdash P_8t_8 \otimes !tn}$$

$$\frac{}{t_z \vdash (P_4t_4 \oplus P_5t_5 \oplus P_6t_6 \oplus P_7t_7 \oplus P_8t_8) \otimes !tn} \ R\oplus, !c$$

$$\frac{}{t_x \vdash (P_9t_9 \oplus P_{10}t_{10} \oplus P_{11}t_{11} \oplus P_{12}t_{12}) \otimes !tn} \ R\oplus, !c$$

$$\frac{}{t_y \vdash (P_1t_1 \oplus P_2t_2 \oplus P_3t_3) \otimes !tn} \ R\oplus, !c$$

5.0 Impact on Requirements Engineering

The related work is on the motion of arm conversion into Petri NETS as we know that the Petri NETS are the graphical representation and through graph we can easy understand the problem. If we use the graph for our requirements it will cause very clear requirements gathering and when we have pure and neat requirements farther it is easy to build a product. Farther it will decrease the chances of inaccuracy that may be able to cause by the requirements gathering.

6.0 Conclusion

In our experience it is not possible for each and every one that has full command on physics and math proof, the shortest way to find and to observe the problem is graphical representation of an object. That's why it is very easy approach to find the problem and calculate them. Thought this paper may we will able to prove other motions on simple way.

References

DiCesare F, Harhalakis G, Proth JM, et al. 1993. Practice of Petri NETS In Manufacturing. Rensselaer Polytechnic Institute, Troy, USA

Kim SS, Kim CH, Park JH. 2006. Human-like Arm Motion Generation for Humanoid Robots Using Motion Capture Database. 133-791, School of Mechanical Engineering, Hanyang University, Seoul, Korea

Kim CH, Kim SS, Ra SK, et al. 2007. Regenerating Human-like Arm Motions of Humanoid Robots for a Movable Object. 130-650, Center for Cognitive Robotics Research Korea Institute of Science and Technology. Seoul, Korea

Lumelsky VJ. 1991. Department of Mechanical Engineering University of Wisconsin, Madison, WI, USA