2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Formal Verification of a Collision-free Algorithm of Dual-arm Robot in HOL4

Liming Li[1] ,Zhiping Shi[2], Yong Guan[3], Chunna Zhao[4], Jie Zhang[5] and Hongxing Wei[6]

Abstract— **Possessing two manipulators heightens the ability of dual-arm robots (DAR) to conduct complex tasks, while raising hazard that the two manipulators might collide with each other or with other objects. DARs are usually equipped with a collision-free motion planning algorithms (CFMPA) to prevent the two manipulators from colliding. The CFMPA searches the motion paths of robot manipulators, which are expected to be as short and smooth as possible under the premise of ensuring safety. It is important to ensure that the algorithm is correct and efficient. It is not enough to apply traditional test methods to determine whether DARs can work in safety-critical applications. In this paper, theorem proving technology is employed to analyze the correctness and efficiency of a classical CFMPA. The CFMPA is outlined, and then formalized in high order logic with the theorem prover HOL4. An inconsistency in the range of motions of the robot manipulators in the algorithm is discovered. An improved algorithm is therefore proposed. Formal verification with HOL4 proves the correctness and efficiency of the proposed algorithm that has already run on a real DAR as well, in conformity with our expectation.**

I. INTRODUCTION

As scope of application of robots grows, the issue of safety has become paramount. DARs can perform many kinds of work requiring dexterity. Such work is executed by dual manipulators. When a DAR is at work, there are two manipulators moving simultaneously in the working area. How to make the two manipulators cooperate with each other and avoid colliding becomes a key problem. So CFMPA has been proposed to address the problem. Now the new problem is to verify the correctness and the efficiency performance of the CFMPA of dual arms robots.

Generally, designers try to ensure correctness by simulation and testing. The algorithm verified in this paper was simulated when it was designed [1]. These methods of verification are helpful and easy to use but not enough because they cannot satisfy the security requirements in relevant standards such as IEC 61508 [2]. Safety of algorithms is established by the safety requirements, and the conformance of the algorithm designs or implementations with these must be verified [7]. Formal methods are recommended in some standards like IEC 61508 [2]. Formal verification provides the mathematically rigorous, machine-checked proof of the correctness of a program with respect to the safety requirements. There are two main methods of formal verification. One is model checking, which is widely used in many research institutes to verify the motion planning of robots. Jones et al. verified a receding horizon algorithm for informative path planning with temporal logic constraints [3]. Kress-Gazit et al. provided a framework to automatically generate a hybrid controller that guarantees that the robot can achieve its task when a robot model, a class of admissible environments, and a high-level task or behavior for the robot are provided [1]. Although model checking possesses a high degree of automation, it is restricted by the scale of the verified question and the expressive ability because the state space explosion is inevitable and some complex problem cannot be represented accurately. The other method of formal verification is theorem proving, which has a strong expressive ability to verify the correctness of a particular algorithm, especially in high order logic. Meikle proved Graham's scan algorithm in Isabelle's Hoare logic [5]. Taubig verified the safety area algorithm of an autonomous robot in Isabelle/HOL [6]. To the best of our knowledge, this is the first time that the collision-free algorithm of a DAR in HOL4 has been verified.

From a theorem proving perspective, CFMPA algorithms are challenging, because they involve a lot of geometry modeling the behavior of robot manipulators so sophisticated mathematical domain model must be built in logic. Mathematically, geometry is concerned with the properties of subsets of $\mathbb{R}^2$ or $\mathbb{R}^3$. To capture these concepts formally, expressive forms of logic such as set theory or higher-order logic are required, as they allow us to formalize mathematics involving numbers, lists, sets, vectors, matrix, and functions. We use a general-purpose theorem prover built on higher-order logic, HOL4, as the verification tool for our robotics application.

*This work was supported by the International S&T Cooperation Program of China (2010DFB10930, 2011DFG13000); the National Natural Science Foundation of China (60873006, 61070049, 61170304, 61104035); the Beijing Natural Science Foundation and S&R Key Program of BMEC(4122017, KZ201210028036); the Open Project Program of State Key Laboratory of Computer Architecture and the Open Project Program of Guangxi Key Laboratory of Trusted Software.

[1]Liming Li is with the Beijing Key Laboratory of Electronic System Reliability Technology, Capital Normal University, 100048 Beijing, China lilimingа@126.com

[2]Zhiping Shi is corresponding author and with the Beijing Key Laboratory of Electronic System Reliability Technology, Capital Normal University, 100048 Beijing, China shizhiping@gmail.com

[3]Yong Guan is with the Beijing Key Laboratory of Electronic System Reliability Technology, Capital Normal University, 100048 Beijing, China guanyxxxy@263.net

[4]Chunna Zhao is with the Beijing Key Laboratory of Electronic System Reliability Technology, Capital Normal University, 100048 Beijing, China chunnazhao@163.com

[5]Jie Zhang is with College of Information Science & Technology, Beijing University of Chemical Technology, 100029 Beijing, China jzhang@mail.buct.edu.cn

[5]Hongxing Wei is with the School of Mechanical Engineering and Automation, Beihang University,100083 Beijing, China weihongxing@buaa.edu.cn

HOL4 developed in Cambridge University is the latest version of the HOL interactive proof assistant for higher order logic. By the theorem prover, simple proof steps could be run automatically, and users can concentrate on the hard parts. The prover has mechanisms to allow for a high degree of confidence of the correctness of proofs than paper-and-pencil proof methods do. Moreover the theorem proving method requires users to deeply understand designs or implementations and specifications of systems. This is helpful to uncover hidden faults which are apt to be missed by tests and simulations. To describe the algorithm, we have developed required foundation theorem libraries including vector & matrix theories, function matrix theory, the Hoare logic, separate logic and the weakest pre-predicate theorems of Dijkstra in HOL4.

This paper is structured as follows. In Section II, we sketch out the actual collision-free algorithm of a DAR. In Section III, the formal domain is modeled and our approach of specification and verification is given. In Section IV, a semantic inconsistency in the algorithm is discovered, and an improved method is provided and verified using a formal method.

II. CFMPA FOR DARS

A CFMPA for a DAR is a collision avoidance algorithm, which protects the two manipulators of the robot, allowing for simultaneous movement in a plane without collision. In this paper, a 2D horizontally articulated DAR (SCARATES robot) [1], which is shown in Fig 1, is chosen as an instance for the algorithm. The algorithm can also be applied to other types of DARs. Each of the manipulators has two joints.

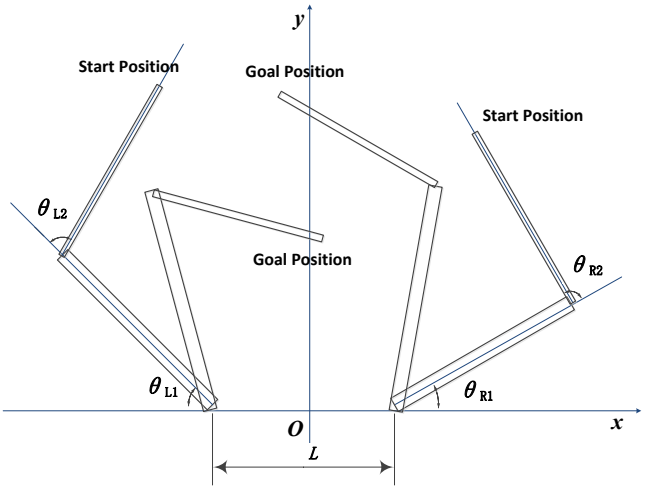


Fig. 1: The configuration structure of SCARATES

In robotics, the positions and orientations of rigid objects are applied in a physical space are formulated as coordinate parameters of a configuration space (C-space) which is, in essence, a general space. The C-space of a rigid object in real life has six degrees of freedom (d.o.f), three of which describe the position of the rigid object, while the other three describe the object's orientation. A rigid object that has a specified position and orientation in physical space will be converted to a specified point in the configuration space. The C-space can be further divided into two subspaces: obstacle space and free space. Obstacle space refers to the parts of the C-space occupied by obstacles, and free space refers to the parts of the C-space not occupied by obstacles. The C-space is usually used for planning the motions of a rigid object among stationary obstacles. However, in our work, we adapt the method of C-space for collision-free motion planning of DARs. The two manipulators form each other's obstacle spaces in CFMPA.

The coordinates of the C-space are the angles of two joints of the subordinate manipulator, and are noted as $(\theta_{R1}, \theta_{R2})$. Suppose that the range of the angle of the joints is limited to between 0 and 2π, and that the C-space of the subordinate manipulator is expressed as: $C_R = \{(\theta_{R1}, \theta_{R2}) | 0 \leq \theta_{R1} \leq 2\pi, 0 \leq \theta_{R2} \leq 2\pi\}$. The left manipulator is selected as the principal and the right as the subordinate. The limited rotational velocities of the four joints of the robot are ω_{R1}, ω_{R2}, ω_{L1}, ω_{L2} respectively. The principal manipulator moves by PTP from the start position $(\theta_{SL1}, \theta_{SL2})$ to the goal position $(\theta_{GL1}, \theta_{GL2})$. The performance period is computed by $T_f = MAX(|\theta_{GL1} - \theta_{SL1}|/\omega_{L1}, |\theta_{GL2} - \theta_{SL2}|/\omega_{L2})$. This algorithm is aimed at planning a path for the subordinate manipulator in C-space. The overall algorithm is given in Algorithm 1.

Algorithm 1. The original CFMPA

Input:
performance period T_f;
latency Δt; //time interval
the start position $(\theta_{SR1}, \theta_{SR2})$ and the goal position $(\theta_{GR1}, \theta_{GR2})$ of the subordinate manipulator's two joints.

Algorithm: For the sake of presenting its structure, we omit its details and abstract it as follow.
step 1:$N = \frac{T_f}{\Delta t}$; //discrete T_f to N segments
step 2 : $m = \frac{2\pi}{\omega_{R1} \cdot \Delta t}$; $n = \frac{2\pi}{\omega_{R2} \cdot \Delta t}$;
//discrete C-space to m*n parts
step 3: //S_{LArm} from $(\theta_{L1} - \omega_{L1} \cdot \Delta t, \theta_{L2} - \omega_{L2} \cdot \Delta t)$ to $(\theta_{L1} + \omega_{L1} \cdot \Delta t, \theta_{L2} + \omega_{L2} \cdot \Delta t)$;
the maximum area moved by the left manipulator in Δt.
S_{RArm} from $(\theta_{R1} - \omega_{R1} \cdot \Delta t, \theta_{R2} - \omega_{R2} \cdot \Delta t)$ to $(\theta_{R1} + \omega_{R1} \cdot \Delta t, \theta_{R2} + \omega_{R2} \cdot \Delta t)$;
the maximum area moved by the right manipulator in Δt.
//build the database of collision state D_R.
builtDR;
//Define builtDR as follows.
{For all k=0 to N do
For all i=0 to m do
For all j=0 to n do
if SLARM[k] ∩ SRARM[k][i][j]=Φ
then $P_{k,i,j}$ =0 else $P_{k,i,j}$=1;
//$P_{k,i,j}$ is the weight of pint (k,i,j)
}

step 4: //Search a path L from the start position to the goal position in D_R.
searchoptpath(startp,goalp);
//linear program a optimal path base on the minimum cost function J1
$J1(s) = \sum_{k=1}^{N} (|\Delta i| + |\Delta j|)$(the shortest movement distance)
if there are multiple minimum J1s, base on the minimum cost function J2
$J2(s) = \sum_{k=1}^{N} (|\Delta\Delta i| + |\Delta\Delta j|)$(the smoothest movement)

Output and guarantees. If the robot satisfies the assumptions described above and the input parameters are correct, the algorithm guarantees the safety and correctness of the optimal path. This means that the two manipulators of the robot will always be able to move from the start point to the goal point without collisions. More precisely, no part of the manipulators will intersect.

III. FORMAL MODELING AND VERIFICATION

There are three components of our formalizing framework for the CFMPA for DAR: the position of the joints, the C-space obstacle boundary, and the structure of the algorithm. All three components can be manipulated in the HOL4 theorem prover. We use the function matrix and vector theory to represent the position of the joints, and set theory to represent the C-space obstacle boundary. These two theories as a whole are referred to as domain theory. In addition, we use Hoare logic and the weakest pre-predicate of Dijkstra to represent the structure of the algorithm. Then, we verify the collision avoidance and function correctness in HOL4 in a straightforward manner, based on our formalizing framework in Fig.2.

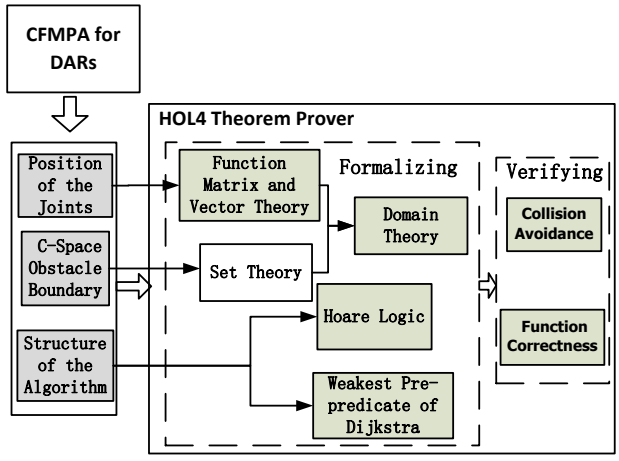


Fig. 2: Overview of the process of modeling and verifying

This algorithm has two key properties: the safety and the functional correctness. The safety of the algorithm represents that the obstacle space of the principal manipulator and subordinate manipulator cannot be overlapped at any time. In the CFMPA, S_{LArm} and S_{RArm} denotes the obstacle space of the principal manipulator and the subordinate manipulator at the time of t respectively, while $(\theta_{R1}, \theta_{R2})$ characterizes the position of the subordinate manipulator. The functional correctness means that the algorithm can find an optimal path, shortest and smoothest, if the safety property can be satisfied when the subordinate manipulator move from the starting position to the goal position and otherwise returns the failure message of the path searching. Meanwhile, we must ensure that the algorithm can be terminated.

In order to formalize the obstacle space, we use the convex polygon model to describe the S_{LArm} and S_{RArm} so that we can employ intersection of the polygon to determine whether the obstacle space is overlapped or not.

A. Convex Polygon Domain Modeling

For the space on which the algorithm is based is a subset of $\mathbb{R}^2$, we can specify the point of the space in HOL4 as follows.

```
val _=type_abbrev("Point",``:real[2]``);
```

So, many definitions and theorems concerning basic geometry can be directly transformed from textbook math to HOL4. For every pair of points among the set called a convex, every point on the line segment that joins them is also in the set. In HOL4:

Definition 1: segment_def:

$$segment(x : Point\ \ y : Point) := \{z : Point | \exists t : \mathbb{R}.\ 0 \le t \le 1 \wedge (z = t * x + (t-1) * y)\}$$

Its type is $Point \to Point \to Point\ set$.

And the convex hull of a set Z is the smallest convex set containing Z:

Definition 2: is_convex_def:

$$is\_convex(Z : Point\ set) := \forall x\ y : Point.\ x \in Z \wedge y \in Z \Rightarrow segment\ x\ y \subseteq Z$$

Its type is $Point\ set \to bool$.

Definition 3: convex_def:

$$convex(X : Point set) := \cap\{Y | is\_convex\ Y \wedge X \subseteq Y\}$$

Its type is $Point\ set \to Point\ set$.

Using these definitions, many simple theorems can be proven by just rewriting definitions.

In general, Graham's Scan is used to compute the convex hull of a more complex set of points and the convex hull represents the area to be described. In the proof of this paper, the following two lemmas are used.

Convex monotone lemma could be formalized as follows:

Lemma 1: convex_monotone:

$$\forall X\ Y (: Point\ set).X \subseteq Y \Rightarrow convex\ X \subseteq convex\ Y$$

Convex union lemma could be formalized as follows:

Lemma 2: convex_union:

$$\forall X\ Y (: Point\ set).(convex\ X \cup convex\ Y) \subseteq convex\ (X \cup Y)$$

The CFMPA only involves a problem of collision avoidance in 2D, so we only need to model the convex polygons. We use $\mathbb{R}^2$ to represent the type of points, hence the point set is a function of $\mathbb{R}^2 \to bool$. But, in practice, the movement

space of the arm robot is $\mathbb{R}^3$. We can easily extend the domain model from 2D to 3D, simply by using $\mathbb{R}^3$ and $\mathbb{R}^3 \rightarrow bool$ to respectively represent the point and point set. Meanwhile the model is a convex polyhedron. In our work, we extend the discussion to the model with the convex polyhedron and use the overlapping of sets to describe the collision of objects in 3D.

B. Verifying the Algorithm with a Formal Method

We verify the CFMPA applying Hoare logic. Hoare triples are written as $\{P\}\ C\ \{Q\}$, where P and Q are assertions and C is a program. In the CFMPA, a pair of collision-free optimal paths will be automatically generated when the start position, goal position and orientations of the two manipulators of the DAR are given. There are two main steps to planning a collision-free path: (1) confirm the database of the collision-free state D_R; (2) search the optimal collision-free path from D_R.

The function of step 3 in the CFMPA is confirming the database of the collision-free state D_R in Fig 3. Each of the states in D_R in Fig.3 is determined by whether the obstacle spaces of the two manipulators are overlapping. Each manipulator is simplified as the merger of two simple convex polygons of its two joints, therefore the obstacle space of one manipulator is the merged polygon of other manipulator.

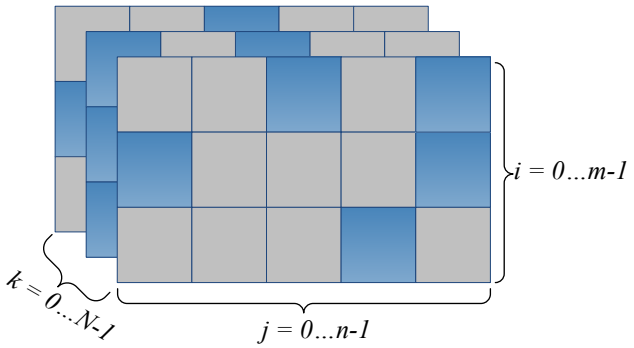


Fig. 3: The kth layer corresponds to the states of the all the positions at each time of Δt, which is nominated as $D_{R,k\Delta t}$. The state of the positions that correspond to light-colored grid is 0, while state of the rest is 1. The state database D_R is constructed by all the layers $D_{R,k\Delta t}$ at each time of Δt, which is $D_R = \bigcup_{k=0}^{N-1} D_{R,k\Delta t}$.

Consider the Hoare triple:

Theorem 1:

$$\{T\}$$
$$BuiltDR$$
$$\{\forall(i\ j\ k) : \mathbb{N}.\ k < N \wedge i < m \wedge j < n \Rightarrow$$
$$P_{i,j,k} = ((SLarm_k \cap SRarm_{i,j,k}) = \Phi \rightarrow 0|1)\}$$

where N is the determined natural numbers and m, and n are the numbers that indicate that the C-space is divided. $(t \rightarrow t_1|t_2)$ means that if t then t_1 else t_2.

Based on the Hoare rules, Theorem 1 can easily be proven. Therefore, Property 1 can also be proven.

Property 1:

$$\forall(i\ j\ k) : \mathbb{N}.\ k < N \wedge i < m \wedge j < n \Rightarrow$$
$$(P_{i,j,k} = 0 \Rightarrow (SLarm_k \cap SRarm_{i,j,k}) = \Phi) \wedge$$
$$(P_{i,j,k} = 1 \Rightarrow (SLarm_k \cap SRarm_{i,j,k}) <> \Phi)$$

This means that there is no collision when the state is 0.

The function of step 4 of the algorithm is to search for the optimal collision-free path from D_R. The path is the group of ordered coordinates that represent the position at each respective time. The path can be denoted by *list* type. In order to accurately describe step 5 in Hoare logic, we designed the following definitions.

Whether a *list* is a path is defined as follows.

Definition 4: is_path_def:

$$is\_path(L : list, startp, goalp) :=$$
$$(L_0 = startp) \wedge (L_{LENGTH(L)-1} = goalp) \wedge$$
$$\forall(i\ j\ k\ i'\ j') : \mathbb{N}.$$
$$i < m \wedge i' < m \wedge j < n \wedge j' < n \wedge k < LENGTH(L) \Rightarrow$$
$$L_k = (i, j) \wedge L_{k+1} = (i', j')$$
$$\Rightarrow (i' = i \vee i' - i = 1 \vee i - i' = 1) \wedge$$
$$(j' = j \vee j' - j = 1 \vee j - j' = 1) \wedge$$
$$(k < N \Rightarrow P_{k,i,j} = 0) \wedge (k \geq N \Rightarrow P_{N-1,i,j} = 0)$$

The optimal path is the one with the smallest total distance of movement and the smoothest direction. $J1$ and $J2$ are the cost function in the algorithm.

Definition 5: optpath_def:

$$optpath(startp, goalp) := \varepsilon LA.$$
$$is\_path(LA) \wedge$$
$$(\forall L.\ is\_path(L) \Rightarrow J1(LA) \leq J1(L) \wedge$$
$$(J1(LA) = J1(L) \Rightarrow J2(LA) \leq J2(L)))$$

where $\varepsilon x.\ t$ means that an x such that: t.

Then, the Hoare triple of step 4 can be expressed as Theorem 2.

Theorem 2:

$$\{\forall(i\ j\ k) : \mathbb{N}.\ k < N \wedge i < m \wedge j < n \Rightarrow$$
$$P_{i,j,k} = ((SLarm_k \cap SRarm_{i,j,k}) = \Phi \rightarrow 0|1)\}$$
$$searchoptpath$$
$$\{\forall(i\ j\ k) : \mathbb{N}\ L : list\ startp\ goalp.$$
$$optpath <> Null \Rightarrow$$
$$(i < m \wedge j < n \wedge k < N \Rightarrow$$
$$optpath(startp, goalp) = L \wedge L_k = (i, j) \Rightarrow P_{k,i,j} = 0) \wedge$$
$$(i < m \wedge j < n \wedge N \leq k < LENGTH(L) \Rightarrow$$
$$optpath(startp, goalp) = L \wedge L_k = (i, j) \Rightarrow P_{N-1,i,j} = 0)\}$$

Based on the definition of **is_path_def**, the properties of path can be captured.

Since the path must start from the starting position and end at the goal position, this property can be formalized as follows.

Property 2:

$$\forall L : list\ startp\ goalp.$$
$$is\_path(L, startp, goalp) \Rightarrow$$
$$(L_0 = startp) \wedge (L_{LENGTH(L)-1} = goalp)$$

In the path generated by the algorithm, each position of the path is remarked as state 0. This property can be formalized as follows.

Property 3:

$$\forall(i\ j\ k):\mathbb{N}\ L: list\ startp\ goalp.$$
$$i<m\wedge j<n\wedge k<N\Rightarrow$$
$$is\_path(L,startp,goalp)\wedge L_k=(i,j)\Rightarrow P_{k,i,j}=0$$

If the principal manipulator reach the goal position while the subordinate manipulator do not reach the goal position yet, the obstacle space of the subordinate manipulator will remain unchanged. Thus, we only need to search in the state database of the last layer in Fig 3. Meanwhile, each position of the path is remarked as state 0. This property can be formalized as follows.

Property 4:

$$\forall(i\ j\ k):\mathbb{N}\ L: list\ startp\ goalp.$$
$$i<m\wedge j<n\wedge N\le k<LENGTH(L)\Rightarrow$$
$$is\_path(L,startp,goalp)\wedge L_k=(i,j)\Rightarrow P_{N-1,i,j}=0$$

Property 3 and property 4 show that the state of every position in the path is 0.

Therefore, we can prove that the optimal path from the start position to the goal position must be the path according to the definition of **optpath_def**.

Property 5:

$$\forall startp\ goalp.$$
$$is\_path(optpath(startp,goalp),startp,goalp)$$

Therefore, the state of the position in the optimal path must be 0.

Property 6:

$$\forall(i\ j\ k):\mathbb{N}\ L: list\ startp\ goalp.$$
$$i<m\wedge j<n\wedge k<N\Rightarrow$$
$$optpath(startp,goalp)=L\wedge L_k=(i,j)\Rightarrow P_{k,i,j}=0$$

Property 7:

$$\forall(i\ j\ k):\mathbb{N}\ L: list\ startp\ goalp.$$
$$i<m\wedge j<n\wedge N\le k<LENGTH(L)\Rightarrow$$
$$optpath(startp,goalp)=L\wedge L_k=(i,j)\Rightarrow P_{N-1,i,j}=0$$

Theorem 2 can be proven on the basis of the above theorems. It shows that all of the positions in the optimal path searched by step 4 from D_R are collision-free. Moreover, the result of the searching is the optimal path or a null *list*, and the null *list* shows that there is no path from the start position to the goal position.

Therefore, we can verify the path generated by the CFMPA according to the above-mentioned properties. Based on the HOL4 theorem prover, the correctness of the properties has been proved. The proving process reveals that the path generated by the CFMPA could be collision-free and optimal if the control accuracy of the robot system is ensured.

C. A Semantic Inconsistency

Although the result of the proof demonstrates that the path generated by the CFMPA can avoid the collision of the manipulators, we find from the proof that the theorem 2 does not need such a strong premise, which will lead to an inefficient algorithm. The fundamental reason is that there exists a semantic inconsistency in the CFMPA.

At the beginning of the algorithm, the path of the principal manipulator is already planned and its direction of movement is determined at each moment. However, when planning the path of the subordinate manipulator, the algorithm considers that each joint of the principal manipulator has two possible directions of rotation: the clockwise and counterclockwise. Therefore the domain that the principal manipulator will go through in the next Δt, noted as S_{LArm}, is limited to that between the position $(\theta_{L1}-\omega_{L1}\cdot\Delta t,\theta_{L2}-\omega_{L2}\cdot\Delta t)$ and $(\theta_{L1}+\omega_{L1}\cdot\Delta t,\theta_{L2}+\omega_{L2}\cdot\Delta t)$.

In fact, the rotation direction of the principal manipulator is known. That is to say, the area that the principal manipulator will sweep into in the next Δt, noted as S'_{LArm}, is a part of S_{LArm}. If the rotation directions of two joints of the principal manipulator are both clockwise, S'_{LArm} is limited to between the position $(\theta_{L1},\theta_{L2})$ and $(\theta_{L1}+\omega_{L1}\cdot\Delta t,\theta_{L2}+\omega_{L2}\cdot\Delta t)$. This is an inconsistency relating to the range of movements of the principal manipulator in the next Δt. Such an inconsistency in the design of the algorithm has potential risks. Performance could be greatly reduced and security could be threatened.

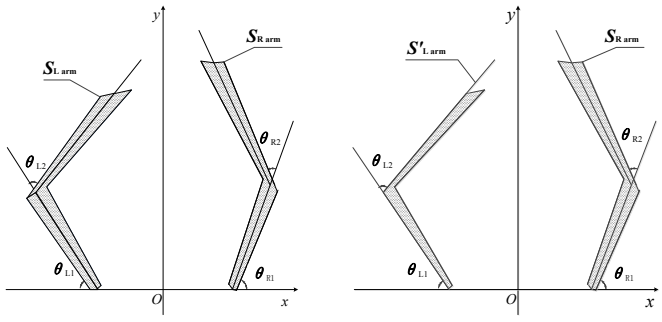
IV. VERIFYING THE MODIFIED ALGORITHM

A. Modifying the Algorithm

Firstly, according to the consistent requirement, we modify the original algorithm. At every interval, the rotation direction of the principal manipulator can be read from its planned path. So S'_{LArm} is computed within the ranges of domain.

$$\begin{aligned}&if\ \theta_{GL1}>\theta_{SL1}\ then\\&\ \{if\ \theta_{GL2}>\theta_{SL2}\ then\\&\quad S'_{LArm}=f((\theta_{L1},\theta_{L2}),(\theta_{L1}+\omega_{L1}\cdot\Delta t,\theta_{L2}+\omega_{L2}\cdot\Delta t))\\&\ else\ S'_{LArm}=f((\theta_{L1},\theta_{L2}),(\theta_{L1}+\omega_{L1}\cdot\Delta t,\theta_{L2}-\omega_{L2}\cdot\Delta t))\}\\&\ else\\&\ \{if\ \theta_{GL2}>\theta_{SL2}\ then\\&\quad S'_{LArm}=f((\theta_{L1},\theta_{L2}),(\theta_{L1}-\omega_{L1}\cdot\Delta t,\theta_{L2}+\omega_{L2}\cdot\Delta t))\\&\ else\ S'_{LArm}=f((\theta_{L1},\theta_{L2}),(\theta_{L1}-\omega_{L1}\cdot\Delta t,\theta_{L2}-\omega_{L2}\cdot\Delta t))\}\end{aligned}$$

Secondly, we can get the new domain S'_{LArm}, as the shadow shown in Fig 4(b). In this light, S'_{LArm} is roughly equivalent to the half of the original domain S_{LArm}, as the shadow is shown in Fig 4(a). Thus, the choices of the path are more flexible than before when planning the path.



(a)before the modification (b)after the modification

Fig. 4: the change of the obstacle space of the subordinate manipulator

B. The Improvement of Performance

Similar to the verification of the original algorithm, we use the Hoare triples to specify the modified algorithm.

Theorem 3:

$$\{T\}$$
$$BuiltDR$$
$$\{\forall(i\ j\ k):\mathbb{N}.\ k<N\wedge i<m\wedge j<n\Rightarrow$$
$$P_{i,j,k}=((SLarm'_k\cap SRarm_{i,j,k})=\Phi\rightarrow 0|1)\}$$

Theorem 4:

$$\{\forall(i\ j\ k):\mathbb{N}.\ k<N\wedge i<m\wedge j<n\Rightarrow$$
$$P_{i,j,k}=((SLarm'_k\cap SRarm_{i,j,k})=\Phi\rightarrow 0|1)\}$$
$$searchoptpath$$
$$\{\forall(i\ j\ k):\mathbb{N}\ L:list\ startp\ goalp.$$
$$optpath<>Null\Rightarrow$$
$$(i<m\wedge j<n\wedge k<N\Rightarrow$$
$$optpath(startp,goalp)=L\wedge L_k=(i,j)\Rightarrow P_{k,i,j}=0)\wedge$$
$$(i<m\wedge j<n\wedge N\leq k<LENGTH(L)\Rightarrow$$
$$optpath(startp,goalp)=L\wedge L_k=(i,j)\Rightarrow P_{N-1,i,j}=0)\}$$

The proving of these two theorems shows that the safety and functional correctness of the modified algorithm is guaranteed.

Base on the analysis of the Section IV.A, $S'_{LArm}\subseteq S_{LArm}$ can be obtained. So the following property can be deduced.

Property 8:

$$(\forall(i\ j\ k):\mathbb{N}.\ k<N\wedge i<m\wedge j<n\Rightarrow$$
$$P_{k,i,j}=((SLarm_k\cap SRarm_{k,i,j})=\Phi\rightarrow 0|1))\Rightarrow$$
$$(\forall(i\ j\ k):\mathbb{N}.\ k<N\wedge i<m\wedge j<n\Rightarrow$$
$$P_{k,i,j}=((SLarm'_k\cap SRarm_{k,i,j})=\Phi\rightarrow 0|1))$$

Based on the precondition strength rule in Hoare logic, Theorem 2 can be proven more easily.

Pre-strength rule:

$$\frac{|-P'\Rightarrow P\quad |-\{P\}C\{Q\}}{|-\{P'\}C\{Q\}}$$

Theorem 2 and 4 show the precondition of the modified algorithm is weaker than the original. This means that the modified algorithm can have higher efficiency. This conclusion is drawn as follows. With the modified algorithm, there are more states the weights of which are 0 in the database D'_R, so the optimal path is searched more easily; moreover the cost of the manipulator's path is lower.

The improved algorithm has been tested in our experimental platform. The experiment shows that it is more efficient than the previous algorithm. In addition, the improved algorithm succeeds in searching especially in the conditions where the previous one fails.

V. CONCLUSIONS

We presented a formal method to verify the CFMPA of a DAR. A semantic inconsistency in the algorithm was discovered. In addition, we present a modified algorithm with the improvement of searching efficiency and planning success. That the modified algorithm is collision-free and is more efficient, was verified using in HOL4.

In conclusion, we believe that formally proving the collision-free algorithm in a theorem prover like HOL4 adds confidence in its correctness and we believe that this should become an important stage in the process of developing such algorithms. Despite the difficulty of proving these algorithms at present, we believe that the task will become easier as libraries of useful theories are built and a better understanding of the field is gained. Our work demonstrates how successful Hoare logic can be for formalizing collision-free algorithms. Not only does Hoare logic allow the formal specifications to resemble the algorithm, but it forces one to think carefully about algorithms. Formalizing the properties of the algorithm makes it easier to obtain a better understanding of it and reveals situations that might have gone undetected. In this algorithm, an inconspicuous inconsistency was revealed.

ACKNOWLEDGMENT

First and foremost we thank Prof. Shengzhen Jin for the guidance and encouragement that he gave us. We also thank Prof. Jindong Tan and Prof. Xiaoyu Song for their many good suggestions.

REFERENCES

[1] F. Ding, W. Han ,and X. Zhao, On Collision-Free Motion Planning of a Dual-Arm Robot, Mechanical Science and Technology 21.6 (Nov. No 2002): 930-933

[2] S. Brown, Overview of IEC 61508, Design of electrical/electronic/programmable electronic safety-related systems, Computing & Control Engineering Journal 11.1 (2000): 6-12.

[3] A. Jones , M. Schwager ,and C. Belta, Technical Report: A Receding Horizon Algorithm for Informative Path Planning with Temporal Logic Constraints, arXiv preprint arXiv:1301.7482(2013).

[4] H. Kress-Gazit, G. E. Fainekos , and G. J. Pappas, Temporal-logic-based Reactive Mission and Motion Planning, IEEE Transactions on Robotics, 25.6(2009): 1370-1381.

[5] L. I. Meikle and J. D. Fleuriot, Mechanical Theorem Proving in Computational Geometry, Automated Deduction in Geometry, Springer Berlin Heidelberg, (2006): 1-18.

[6] H. Täubig, U. Frese, C. Hertzberg, et al., Guaranteeing Functional Safety: Design for Provability and Computer-aided Verification, Autonomous Robots,32.3 (2012): 303-331.

[7] D. Walter, H. Täubig, and C. Lüth, Experiences in Applying Formal Verification in Robotics, Computer Safety, Reliability, and Security, Springer Berlin Heidelberg, (2010): 347-360.

[8] M. Norrish, C formalised in HOL, University of Cambridge, Computer Laboratory (1998).

[9] C Lüth and D. Walter, Certifiable Specification and Verification of C Programs, FM 2009: Formal Methods, Springer Berlin Heidelberg(2009): 419-434.

[10] L. Cellier, P. Dauchez , R. Zapata, et al., Collision Avoidance for a Two-arm Robot by Reflex Actions: Simulations and Experimentations, Journal of Intelligent and Robotic Systems, 14.2(1995): 219-238.

[11] H. Tuch, Formal verification of C systems code, Journal of Automated Reasoning, 42.2-4 (2009): 125-187.

[12] N. Vahrenkamp, D. Berenson, T. Asfour, et al., Humanoid motion planning for dual-arm manipulation and re-grasping tasks, Intelligent Robots and Systems, 2009, IROS 2009, IEEE/RSJ International Conference on. IEEE, 2009: 2464-2470.

[13] E. Cohen, M. Dahlweid, M. Hillebrand, et al., VCC: A Practical System for Verifying Concurrent C, Theorem Proving in Higher Order Logics, Springer Berlin Heidelberg, (2009): 23-42.

[14] M. Kloetzer and C. Belta, A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications, Automatic Control, IEEE Transactions on, 53.1 (2008): 287-297.

[15] M. Choi, S. Lim, J. Lee, et al., Cooperative Path Planning for Redundant Dual-Arm Robot Using Low-Dimensional Sample-Based Algorithm, Mechatronic Systems. (2010): 701-708.