ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

MÔ HÌNH HÓA TOÁN HỌC (CO2011)

Đề bài tập lớn

"Đặc tả Smart Contract bằng Linear Logic"

GVHD:	Nguyễn An Khương	
	Huỳnh Tường Nguyên	
	Trần Văn Hoài	
	Lê Hồng Trang	
	Trần Tuấn Anh	
SV thực hiện:	Nguyễn Lê Chí Bảo	1610179
	Nguyễn Đức Duy	1610468
	Nguyễn Ngọc Hoàng	1611160
	Đường Quang Huy	1611244
	Bùi Anh Nhật	1612377

Tp. Hồ Chí Minh, Tháng 6/2018

Mục lục

1 Bài toán 1

1.1 Lịch sử của hợp đồng thông minh (smart contracts)

Cụm từ "smart contacts" (hợp đồng thông minh) được đặt ra bởi nhà khoa học máy tính và mật mã học Nick Szabo năm 1994 và đã được nghiên cứu qua nhiều năm. Nguyên lý hoạt động của nó được Szabo mô tả năm 1996 trong bài báo có tiêu đề "Smart Contracts: Building Blocks for Digital Markets" trên tạp chí Extropy [Sza96], rất lâu trước khi công nghệ blockchain ra đời. Theo ý tưởng của Szabo, smart contracts là những giao thức kỹ thuật số cho việc chuyển dịch thông tin, sử dụng các giải thuật toán học để thực thi các giao dịch một cách tự động một khi các điều khoản được thỏa và dùng để kiểm soát hoàn toàn quá trình này. Định nghĩa đã đi trước thời đại đến hơn 10 năm này vẫn còn đúng cho đến ngày nay. Tuy nhiên vào năm 1996, ý tưởng này vẫn chưa được đón nhận vì các công nghệ cần thiết để vận hành chưa ra đời, cụ thể là sổ cái phân tán (distributed ledger).

Vào năm 2008, Bitcoin, đồng tiền kỹ thuật số hoàn chỉnh đầu tiên, đã được tạo ra dựa trên cơ sở của công nghệ blockchain. Blockchain của Bitcoin không cho phép các điều kiện, để kết thúc 1 giao dịch, được quy định trên một khối (block) mới vì nó chỉ chứa các thông tin giao dịch. Dù sao đi nữa, sự xuất hiện của blockchain đã trở thành nguồn động lực để phát triển smart contracts. Năm 2013, nền tảng blockchain Ethereum đã giúp cho việc đưa smart contracts vào thực tế. Ngày nay, mặc dù thị trường đã cung cấp nhiều nền tảng cho phép sử dụng smart contracts nhưng Ethereum vẫn được sử dụng nhiều nhất.

1.2 Ứng dụng của hợp đồng thông minh

Đây chỉ là một số trong số rất nhiều ứng dụng của hợp đồng thông minh.

1.2.1 Ứng dụng vào bảo hiểm

Smart contract giúp cải thiện trải nghiệm của người dùng cũng như giảm thiểu những sai sót và chi phí trong quá trình giao dịch bảo hiểm. Blockchain cũng như Smart contract tăng tốc quá trình xử lí yêu cầu/ trao trả bảo hiểm, tránh được những lỗi sai sót so với làm bằng tay.
Smart contract cũng giúp tự động hóa việc chuyển tiền đến người sử dụng bảo hiểm và cũng đảm bảo được hợp đồng là hợp lệ. Ví dụ như một người sửa xe của anh ấy chỉ nhận được tiền bảo hiểm nếu anh ta sửa xe tại cửa hàng được chứng nhận và phải có sự xác nhận bởi người thợ sửa xe đó. Một ví dụ khác, smart contract có thể ghi lại các thông tin về thời tiết lên blockchain, sau đó nó đọc thông tin này và trả tiền bồi thường cho chuyến bay của những hành khách mà bị hủy hay bị hoãn do thời tiết xấu. Nó cũng có thể kết hợp với công nghệ IoT và các cảm biến để theo dõi sự hư tổn của các thiết bị trong nhà để có thể tự động đền bù hoặc hỗ trợ chủ nhân ngôi nhà gọi người sửa chữa đến.

1.2.2 Ứng dụng trong chuỗi cung ứng(logistic)

Smart contract giúp giảm chi phí lưu trữ đồng thời cải thiện việc quản lý chuỗi cung ứng tốt hơn, cho phép lưu lại mọi thông tin, thao tác dù là rất nhỏ theo chiều đi của sản phẩm. Nó giúp tăng tính bảo mật, an toàn và rõ ràng trong chuỗi cung ứng. Một khi dữ liệu đã được xác thực và lưu lại thì không thể thay đổi hay đánh cắp nhờ các kết nối thông minh và một số điều kiện mã hóa kèm theo để xác nhận giao dịch là hợp lệ. Doanh nghiệp sẽ nhận được nhiều hơn sự tin tưởng của khách hàng khi cho họ theo dõi quá trình giao dịch, từ đó tăng thêm lợi thế cạnh

tranh cho công ty của mình. Bên cạnh đó, smart contract giúp người quản lý dễ dàng theo dõi và khắc phục lỗi do có sự lưu trữ dữ liệu theo phân cấp thời gian.

1.2.3 Ứng dụng trong việc vay thế chấp

Smart contract giúp tăng thêm tính bảo mật cho người sử dụng để vay mượn tiền, cung cấp chức năng cho phép người cho vay đặt ra các yêu cầu với người vay tiền như là thời gian vay mượn, số tiền cho vay, lãi suất bao nhiêu phần trăm, v.v. Hợp đồng thông minh này có thể tự động và đảm bảo người mượn phải trả đúng thời hạn, nếu không hệ thống sẽ tự động lấy tài sản mà người vay đã thế chấp để bù vào những phần bị trễ hạn cho người cho vay.

1.2.4 Ứng dụng trong hợp đồng việc làm

Loại hình hợp đồng thông minh này giúp cho doanh nghiệp có thể giảm sai sót, chi phí cũng như thời gian trong việc phát lương cho người lao động. Nó cũng đảm bảo người lao động chấp hành tất cả những điều khoản đã đề ra trong hợp đồng, bởi vì hợp đồng thông minh này sử dụng công nghệ Blockchain, không thể bị chỉnh sửa hay lấy cắp. Đồng thời, smart contract cũng sẽ bảo vệ quyền lợi của người lao động, tránh tình trạng bị thiếu lương, mất lương.

1.2.5 Ứng dụng trong bảo vệ nội dung bản quyền

Smart contract ghi lại hoạt động có liên quan đến bản quyền sản phẩm,theo dõi các hoạt động của sản phẩm của mình trên internet, từ đó có thể giúp cho tác giả hay người tạo ra sản phẩm biết được những hành vi vi phạm bản quyền.

1.2.6 Ứng dụng trong bầu cử

Sử dụng hợp đồng thông minh giúp việc bầu cử trở nên công khai nhưng vô cùng bảo mật, an toàn, tránh được sự can thiệp vào kết quả bầu cử. Người bầu cử có thể sử dụng internet để bầu mà không cần phải xếp hàng dài chờ đợi ở các địa điểm bầu cử nhưng vẫn bảo đảm được tính đúng đắn của lá phiếu cũng như tính riêng tư với lựa chọn của họ. Điều này khuyến khích người dân tham gia vào các quyết định có ảnh hưởng đến sự phát triển của xã hội, làm tăng sự tin tưởng của người dân với chính quyền.

1.2.7 Ứng dụng trong lĩnh vực tài chính thương mại

Hợp đồng thông minh có thể cho phép việc chuyển giao hàng hóa quốc tế thông qua thư tín dụng nhanh và thanh toán thương mại, đồng thời cho phép thanh khoản tài sản tài chính lớn hơn. Nó cũng có thể cải thiện hiệu quả tài chính cho người mua, nhà cung cấp và tổ chức.

1.2.8 Ứng dụng trong lĩnh vực y tế, chăm sóc sức khỏe

Trong tương lai, có thể hồ sơ y tế cá nhân sẽ được mã hóa và được lưu trữ trên blockchain với quyền truy cập duy nhất cho một cá nhân cụ thể. Các biên lai cho việc chi trả viện phí, dịch vụ chăm sóc sức khỏe, thuốc thang cũng sẽ được lưu trữ trên blockchain mà từ đó các hợp đồng thông minh có thể tự động kiểm tra và thực hiện việc hoàn phí theo các quy định của bảo hiểm y tế, tai nạn. Nó cũng có thể giúp tự động thực hiện các hỗ trợ tài chính đối với các cá nhân hiến máu, hiến tạng, hay tình nguyện làm các đối tượng nghiên cứu, tham gia các thử nghiệm y học.

1.2.9 Ứng dụng trong bất động sản

Với hợp đồng thông minh, chúng ta sẽ không phải mất phí cho các bên trung gian (môi giới) khi muốn thuê/cho thuê, bán/mua nhà, căn hộ,... Nó sẽ bắt các bên tham gia phải thực hiện đúng các yêu cầu trước thời hạn để thực hiện giao dịch (người thuê phải trả tiền, người cho thuê phải giao mã khóa vào nhà; tiền và khóa được hợp đồng giữ lại chờ khi đến hạn sẽ thực hiện giao dịch). Nếu một trong hai không thực hiện yêu cầu thì hợp đồng bị hủy và những gì hợp đồng đang giữ sẽ được trả cho người gửi tương ứng. Nó cũng giúp người cho thuê tự động quản lý vấn đề tiền thuê nhà (nếu quá hạn mà hợp đồng chưa nhận được tiền thuê, mã khóa nhà sẽ bị thay đổi).

1.2.10 Hợp đồng thông minh trong thời đại IoT

Mạng lưới "vạn vật kết nối Internet" sẽ mang lại cho mọi thiết bị trong cuộc sống hàng ngày khả năng kết nối với Internet, và do đó là khả năng kết nối đến hệ thống blockchain. Từ đó áp dụng hợp đồng thông minh vào blockchain, chúng ta sẽ biến gần như mọi giao dịch, trao đổi hàng ngày thành hoàn toàn tự động nhưng vẫn đảm bảo được tính chính xác của các giao dịch. Các cảm biến ở mọi nơi, hệ thống định vị vị trí,... giúp hợp đồng thông minh dễ dàng xác định được vị trí của gói hàng, kiểm tra được gói hàng đã đến vị trí của người nhận hay chưa, từ đó tự thực hiện các giao dịch hợp lý.

1.3 Lịch sử của "Linear Logic"

Linear logic (LL) (tạm dịch: logic tuyến tính) được giới thiệu bởi Jean-Yves Girard (1947 -) năm 1987. Linear logic (LL) có liên quan chặt chẽ với nhiều logic được xây dựng trước nó. Giáo sư Toán học Joachim Lambek (12/5/1922 – 6/23/2014) đã phát triển nên phép tính Lambek (Lambek calculus), một logic phi giao hoán (noncommutative logic) nhằm để phân tích cấu trúc câu của ngôn ngữ tự nhiên, vài thập kỷ trước khi Girard xây dựng nên LL. Relevance logic và direct logic, đều xuất hiện trước LL tương đối lâu, cũng đã được nghiên cứu khá nhiều khi LL xuất hiện. Một cách đơn giản, có thể nói LL đứng dưới relevance và direct logic, đứng trên Lambek calculus, dựa trên sự bao gồm hay loại trừ một số "quy tắc cấu trúc" ("structural rules") được gọi là làm suy yếu (weakening), làm gọn lại (contraction), và trao đổi (exchange). Direct logic liên quan chặt chẽ với Affine logic (cũng do Girard giới thiệu), là một LL có thêm quy tắc trao đổi không hạn chế (unrestricted exchange rule).

LL phát sinh từ việc nghiên cứu ngữ nghĩa của phép kéo theo (implication) trong intuitionistic logic (tạm dịch: logic trực giác). Girard đã đưa ra hai ngữ nghĩa riêng biệt trong bài viết giới thiệu LL của ông, và việc khám phá các ngữ nghĩa căn bản cho LL vẫn tiếp tục. Lý thuyết chứng minh LL (LL proof) cũng được đưa ra bởi Girard với sự giới thiệu một chuỗi phép tính (sequent calculus) cho LL, và một ký hiệu thay thế dành cho các chứng minh LL được gọi là lưới chứng minh (proof net). Nhiều vấn đề liên quan đến lưới chứng minh hiện nay vẫn đang được nghiên cứu.

1.4 Ứng dụng của "Linear Logic"

Linear Logic có khá nhiều ứng dụng và liên quan mật thiết đến một số vấn đề trong tính toán như lập trình hàm, lập trình logic, lập trình tương tranh, lập trình hướng đối tượng cũng như các vấn đề về logic không đơn điệu trong lập kế hoạch AI và nhiều ứng dụng khác.

Một trong các mảng ứng dụng được phát triển mạnh mẽ chính là lập trình hàm (Functional programming). Curry-Howard isomorphism (tính đẳng cấu) nêu rằng có sự tương đồng giữa những chứng minh (proof) của logic học chủ nghĩa trực quan (intuitionistic logic) và sự tính toán trong ngôn ngữ lập trình hàm. Một chứng minh có thể trở thành một hàm hiệu quả khi có đầy đủ giả thiết và đưa ra được kết luận. Đây chính là ứng dụng được phát triển mạnh mẽ của linear logic và lập trình hàm.

Linear logic cho phép kiểm soát hiệu quả hơn ngữ cảnh và có thể tạo được cả mô hình dữ liệu production lẫn consumption. Điều này mở ra rất nhiều khả năng trong các ứng dụng như lập trình hướng đối tượng (OOP), cơ sở dữ liệu, xử lý ngôn ngữ tự nhiên.

Ngoài ra, linear logic còn được ứng dụng trong nhiều mảng khác nhau như trong Petri Nets, State-Oriented Programming, Chemical Abstract Machine, v.v.

1.5 Chi tiết và ví dụ hàm tuyến tính

Trong logic cổ điển và logic trực quan, nếu có A và $A \to B$ thì có B nhưng A vẫn còn. Điều này chỉ đúng trong toán học nhưng không đúng trong thực tế. Vì thế ta cần một phép implication khác phù hợp hơn. Linear logic đã đưa ra phép linear implication $\multimap$. Biểu thức $A \multimap B$ được hiểu là "chi tiêu A để được B", khi đó biểu thức $A \to B = (!A) \multimap B$.

Ví dụ 1.5.1: Cho A là "tiêu 5000 đồng" và B là "có 1 chai nước suối" thì $A \multimap B$ biểu diễn cho "nếu ta tiêu 5000 đồng thì ta sẽ có 1 chai nước suối nhưng đồng thời ta cũng mất 5000 đồng".

1.6 Chi tiết và ví dụ về phép hội tuyến tính - nhân (linear conjunction - times)

Trong linear logic phép hội tuyến tính được kí hiệu là $\otimes$. Giống với phép $\wedge$, phép $\otimes$ có nghĩa là có cả 2.

Ví dụ 1.6.1: Ta có phương trình hóa học quen thuộc $2H_2 + O_2 \to 2H_2O$ thì trong linear logic phương trình được biểu diễn là:

$$H_2 \otimes H_2 \otimes O_2 \multimap H_2O \otimes H_2O$$

Có 2 H_2 và 1 O_2 thì ra có 2 H_2O nhưng sẽ không còn nhưng chất trước phản ứng nữa.

1.7 Chi tiết và ví dụ về phép hội tuyến tính ("additive conjunction" hoặc "with")

Phép hội tuyến tính "with" là một phép có trong linear logic nhưng không có trong logic cổ điển và được kí hiệu là $\&$. Khác với phép $\wedge$ va phép $\otimes$, phép $\&$ có nghĩa là chọn 1 trong 2 và cho phép chọn là cái này hoặc cái kia.

Ví dụ 1.7.1: Cho A là "có 5000 đồng", B là "có 1 chai nước suối" và C là "có 1 cây bút bi". Ta sẽ có biểu thức $A \multimap B \,\&\, C$ nghĩa là "với 5000 đồng ta có thể có 1 chia nước hoặc 1 cây bút bi, nhưng chỉ 1 trong hai, chai nước hoặc cây bút bi và ta có quyền được chọn".

1.8 Chi tiết và ví dụ về phép tuyển tuyến tính - cộng (linear disjunction - plus)

Trong linear logic, phép đối ngẫu của phép $\&$ là phép tuyển tuyến tính - cộng được kí hiệu là $\oplus$. Giống với phép $\vee$ trong logic cổ điển, phép $\oplus$ có nghĩa là hoặc cái này hoặc cái kia.

Ví dụ 1.8.1: Trong 1 cuộc thi bốc thăm trúng thưởng thì chắc chắnđược 1 trong 2 loại phiếu thăm và phần thưởng tương ứng khi bốc được là "1 chai nước" hay "1 cây bút bi". Cho A là "hành động rút thăm", B là "trúng 1 chai nước suối" và C là "trúng 1 cây bút bi". Khi đó, thì biểu thức $A \multimap B \oplus C$ nghĩa là "khi bốc thăm thì sẽ trúng 1 chai nước hoặc 1 cây bút bi nhưng không biết là cái nào".

1.9 Chi tiết và ví dụ về phép tuyển tuyến tính ("par")

Đối ngẫu với phép $\otimes$ là phép tuyển tuyến tính $\parr$. Phép $\parr$ tương đối khó hiểu. Biểu thức $A \parr B$ có thể được biểu diễn dưới dạng khác là $A^{\perp} \multimap B$ hay $B^{\perp} \multimap A$. Nó có nghĩa là "phải lựa chọn giữa A và B, không được chọn cả 2".

Ví dụ 1.9.1: Ta xét biểu sau:

$$bullet \otimes ShootAt(bullet, target) \multimap HIT(bullet, target) \parr MISS(bullet, target)$$

Theo như ngữ cảnh, khi có 1 viện đạn và bắn viên đạn đó thì viên đạn chỉ có thể trúng hoặc không trúng mục tiêu, không thể có chuyện vừa dính vừa không mục tiêu.

1.10 Chi tiết và ví dụ về phép toán $\perp$

Trong linear logic, phép phủ định tuyến tính $\perp$ giống như trong logic cổ điển. $A^{\perp}$ là phủ định tuyến tính của A, tương tự như phép negation $\neg$ trong logic cổ điển.

Ví dụ 1.10.1: Ta xét các biểu thức sau:

$$(pen)^{\perp} \ (1)$$
$$(!bottle)^{\perp} \ (2)$$

Biểu thức (1) nghĩa là "không có cây bút" và biểu thức 2 nghĩa là "không có vô hạn chai nước". Biểu thức (2) cũng có thể hiểu là có một vài chai nước và có thể được biểu diễn là $?bottle$.

1.11 Chi tiết và ví dụ về phép toán $\top$

Phép toán $\top$ trong linear logic là một hằng. $\top$ có thể được dẫn ra từ bất cứ gì ($A \vdash \top$, $B \vdash \top$, v.v).

Ví dụ 1.11.1: Ta xét các biểu thức sau với A' là "có 10000 đồng", B là "có 1 chai nước suối" và C là "có 1 câu bút bi". Xét biểu thức:

$$A' \multimap (B \otimes C)$$

Biểu thức trên có nghĩa là nêu có 10000 đồng và ta tiêu nó thì ta có thể có được cả 1 chai nước và 1 cây bút bi. Nhưng nếu ta chỉ muốn mua 1 chai nước suối mà không mua cây bút bi thì ta có thể biểu diễn bằng cách:

$$A' \multimap (B \otimes \top)$$

Phép $\top$ ở dây được sử dụng để loại bỏ đi lựa chọn mà ta không mong muốn.

1.12 Chi tiết và ví dụ về phép toán !

Phép ! có tên gọi trong tiếng anh là "of course" hay "bang". Nó được sử dụng để mã hóa mệnh đềnh đúng. biểu thức $!A$ có nghĩa là lặp lại A vô hạn lần. Khi đó biểu thức $!A \multimap B$ sẽ giống với biểu thức $A \Rightarrow B$ trong logic cổ điển vì bây giờ A tuy bị chi tiêu mất đi thì vẫn còn A.

Ví dụ 1.12.1: Cho A là "có 1 tờ tiền 5000 đồng" thì $!A$ sẽ là "có vô hạn tờ tiền 5000 đồng". Ta cũng sẽ có biểu thức $!A \multimap B$ (có vô hạn tờ 5000 đồng sẽ mua được vô hạn chai nước suối).

1.13 Chi tiết và ví dụ về phép toán ?

Trong linear logic cổ điển (với tính đối ngẫu de Morgan), ký tự đối ngẫu de Morgan của "!" được biểu diễn là "?" và được gọi là phép toán "why not". Gần giống với lượng từ "tồn tại $\exists$" và phép ! giống với "với mọi $\forall$ trong logic vị từ. Nếu ! là có vô hạn thì ? sẽ chỉ là có thể có. Biểu thức $?A$ có nghĩa là có thể có A.

Ví dụ 1.13.1: Tiếp tục ví dụ **1.12.1** nhưng lần này ta xét biểu thức $?A \multimap B$. Biểu thức sẽ có nghĩa là "ta có 1 số tờ tiền 5000 đồng nhất định và ta có thể dùng nó để mua nước."

2 Bài toán 2

2.1 Mô tả ngữ cảnh bằng lời

Hiện nay, Game online đang trở thành thể loại game được mọi người yêu thích. Vì vậy, các tựa game luôn phát triển các tính năng sao cho phù hợp với thị hiếu của người chơi. Phần lớn các tựa game online, người chơi sẽ thường sở hữu các vật phẩm (item) có giá trị, được nhận trong quá trình chơi game. Nhưng có một số vật phẩm đó người chơi không sử dụng, và muốn bán lại cho những người chơi khác cần nó. Và kết quả thu được từ việc bán đó chính là tiền ảo trong game, tiền này giúp cho người chơi có thể mua các vật phẩm cần thiết cho bản thân. Tiền ảo này có thể được nạp từ tiền thật... Tuy nhiên, với đại đa số tâm lý người chơi, luôn muốn bán vật phẩm mình với giá cao nhất có thể. Vì vậy đã dẫn đến sự xuất hiện của hệ thống đấu giá sử dụng smart contract để đảm bảo giữa hai người chơi không tin tưởng lẫn nhau có thể giao dịch thành công.

Hệ thống đấu giá sẽ được mô tả như sau: Mỗi vật phẩm (item) thì sẽ có một mã định danh riêng biệt trên toàn server game (chỉ hệ thống mới biết được mã vật phẩm), được cấu tạo là một đoạn mã như sau: Mã vật phẩm + Mã ID người sở hữu. Khi người chơi đem vật phẩm của mình lên đấu giá, đầu tiên, hệ thống sẽ lấy vật phẩm của người chơi và lập tức thay đổi mã sản phẩm (xóa đi đoạn mã ID sở hữu). Lúc đó, vật phẩm sẽ trở thành vật phẩm không có người sở hữu, và được quyền đấu giá.

Khi vật phẩm được đấu giá, người đem vật phẩm đi đấu giá sẽ quy định giá khởi điểm của nó. Thời gian đấu giá sản phẩm sẽ do người đem vật phẩm đi đấu giá quy định, thường là theo số giờ (1 giờ, 2 giờ...) và được hủy việc đấu giá trong thời gian đấu giá. Lúc này, vật phẩm sẽ được đăng lên mục đấu giá để cho các người chơi đấu giá.

Hình thức đấu giá như sau, các người chơi sẽ tiến hành đấu giá sản phẩm với mức giá cao hơn so với người đấu giá cao nhất hiện tại (nếu trường hợp chưa có mức giá của người đấu giá cao nhất hiện tại, thì mặc định sẽ lấy giá khởi điểm mà chủ vật phẩm quy định). Khi người chơi đấu giá vật phẩm, thì số tiền người chơi sẽ mất đi một lượng bằng đúng với số tiền người đó đấu giá (số tiền người chơi đấu giá phải nhỏ hơn hoặc bằng số tiền mà người chơi đó có).

Khi có người đấu giá cao hơn người đấu giá cao nhất, hệ thống sẽ lập tức hoàn trả lại số tiền cho người đấu giá thấp hơn và cập nhật người đấu giá cao nhất, cứ như vậy cho đến khi hết thời gian đấu giá.

Khi hết thời gian đấu giá, có hai trường hợp xảy ra:

- Trường hợp 1: nếu vật phẩm không được ai đấu giá, thì lập tức hệ thống sẽ trả lại vật phẩm cho người sở hữu.
- Trường hợp 2: nếu có người đấu giá thành công, vật phẩm sẽ được hệ thống xử lý mã vật phẩm bằng cách thêm ID người đấu giá thành công vào vật phẩm và gửi về túi đồ của người đấu giá cao nhất, đồng thời người đem vật phẩm đi đấu giá sẽ nhận được tiền ngay lập tức.

2.2 Các điều khoản với ngữ cảnh được mô tả

Tham gia vào các điều khoản có các chủ thể sau:

A: Người đem vật phẩm đi đấu giá
B: Người tham gia đấu giá
C: Người đấu giá cao nhất hiện tại

Article 1: Bên A đưa vật phẩm lên phòng đấu giá và đặt thời gian đấu giá t, với giá khởi điểm là x. Vật phẩm sẽ được smart contract giữ lại trong khoảng thời gian t.
Article 2: Trong khoảng thời gian đấu giá, bên A có quyền hủy việc đấu giá. Smart contract sẽ trả lại vật phẩm cho A. Nếu đã có C, thì sẽ hoàn tiền lại cho C.
Article 3: Bên B sẽ tiến hành đấu giá với giá là y, số tiền đấu giá sẽ được smart contract giữ lại.
Article 4: Khi B đấu giá thành công, thì smart contract sẽ hoàn tiền cho C, và B trở thành C.
Article 5: Số tiền đấu giá của B phải lớn hơn số tiền đấu giá của C tại thời điểm đó hoặc lớn hơn giá khởi điểm x (nếu chưa có ai đấu giá) mà smart contract thông báo.
Article 6: Số tiền đấu giá y không vượt quá số tiền hiện có của bên B.
Article 7: Hết thời gian times, vật phẩm không ai đấu giá, smart contract sẽ hoàn trả vật phẩm cho bên A.
Aritcle 8: Hết thời gian times, nếu vật phẩm có người đấu giá, smart contract sẽ chuyển vật phẩm cho bên C, và chuyển tiền cho bên A.

3 Bài toán 3: Sử dụng các kiến thức về linear logic để đặc tả ngữ cảnh trong Bài tập 2

Các điều khoản được nêu ra ở mục **2.2** như sau:

- B phải đặt số tiền y không vượt quá số tiền hiện có để tham gia đấu giá vật phẩm

$$y \otimes Check(y, B) \multimap bidder$$

- Khi B tham gia đấu giá đầu tiên (Số tiền đấu giá của B là y, phải lớn hơn số tiền ban đầu là x), smart contract sẽ cập nhật B thành C

$$B \otimes Higher(y, x) \multimap Convert(B, C)$$

- Khi B tham gia đấu giá (Giá tiền đấu giá của B là y_B phải lớn hơn giá tiền đấu giá của C là y_C), smart contract tiến hành hoàn tiền cho C, và sau đó chuyển B thành C

$$B \otimes C \otimes Higher(y_B, y_C) \multimap (Wallet_C + y_C) \otimes Convert(B, C)$$

- Smart contract sẽ gửi vật phẩm cho C trong trường hợp đấu giá thành công

$$C \otimes Item \multimap Send(Item, C)$$

- Trường hợp việc đem vật phẩm đi đấu giá không thành công, smart contract sẽ trả vật phẩm lại cho A

$$A \otimes Item \multimap Return(Item, A)$$

- Nếu vật phẩm được đấu giá thành công thì smart contract sẽ tự động chuyển tiền cho A

$$(Wallet_A + y_C) \parr (C \otimes (Wallet_C + y_C))$$

- Trong khi chưa hết thời gian đấu giá times, bên A có quyền hủy hợp đồng. Nếu chưa có người đấu giá C, thì smart contract sẽ hoàn lại vật phẩm cho A

$$Times^{\perp} \otimes Item \otimes C^{\perp} \multimap Return(Item, A)$$

- Trong khi chưa hết thời gian đấu giá times, bên A có quyền hủy việc đấu giá vật phẩm. Nếu có người đấu giá C, thì smart contract sẽ hoàn lại vật phẩm cho A, đồng thời hoàn tiền lại cho C

$$Times^{\perp} \otimes Item \otimes C \multimap Return(Item, A) \otimes (Wallet_C + y_C)$$

- Hết thời gian đấu giá Times , nếu có người đấu giá thì smart contract sẽ chuyển vật phẩm cho C, và chuyển tiền cho A

$$Times \otimes Item \otimes C \multimap (Wallet_A + y_C) \otimes Send(Item, C)$$

- Hết thời gian đấu giá Times, nếu không có người đấu giá thì smart contract sẽ hoàn lại vật phẩm cho A

$$Times \otimes Item \otimes C^{\perp} \multimap Return(Item, A)$$

4 Bài toán 4: Sử dụng mã giả để lập trình smart contract minh họa cho ngữ cảnh được đưa ra trong Bài tập 2

Mã giả để lập trình smart contract minh họa cho ngữ cảnh được đưa ra ở mục **2.1**. Trong bài tập lớn này, mặc định việc hệ thống xử lý mã item của sản phẩm là có sẵn.

Giải thuật 1 Chọn người trả giá cao hơn và hoàn tiền cho người trả giá cao nhất trước đó

Input: highestBidder, highestBid, player, newBiddingValue, timestamp_contract, ended
textbfOutput:

```
if Converting system time to timestamp <= timestamp_contract and boolean(ended) = false) then
    if newBiddingValue > highestBid then
        if exists(highestBidder) then
            transfer money(highestBid) to the highestBidder;
        end
    end
end
```

Giải thuật 2 Giao dịch khi phiên đấu giá kết thúc

Input: owner, highestBidder, highestBid, itemKey, timestamp_contract, ended
textbfOutput:

if *Converting system time to timestamp >= timestamp_contract* ***and*** *boolean(ended) = false)* **then**
    if *exists(highestBidder)* **then**
        Transfer money(highestBid) to owner;
        Transfer code(itemKey) to highestBidder;
        Cancel the possession of the owner to the key;
    end
    else
        Transfer code(itemKey) to owner;
    end
end
else
    Waiting for the auction to end;
end

5 Bài toán 5: Sử dụng Solidity hoặc Go để lập trình smart contract minh họa cho ngữ cảnh trong Bài tập 2

AuctionItem.sol

```
pragma solidity ^0.4.24;

import "./DateTime.sol";
contract AuctionItem
{
    address public owner; // nguoi dem vat pham di dau gia
    uint public auctionEnd; // thoi gian ket thuc
    DateTime dt;

    // Trang thai hien tai cua qua trinh dau gia
    address public highestBidder; // nguoi dau gia cao nhat
    uint public highestBid; // gia dau cao nhat
    uint public defaultBid; // gia dau khoi diem

    // Cho phep nguoi dau gia thap hon rut tien lai
    mapping(address => uint) pendingReturns;
    // Ma vat pham
    mapping(address => string) itemKey;

    // Cho phep nguoi dem vat pham di dau gia huy ban dau gia.
    // Neu cuoc ban dau gia khong bi huy thi se duoc gan gia tri true khi
    // ket thuc phien dau gia nham khong cho phep su can thiep sau thoi han
    .
    bool ended = false;

    // Cac su kien trong phien dau gia
```

```
26      /// Gia dau tang!
27      event HighestBidIncreased(address bidder, uint amount);
28      /// Ket thuc!
29      event AuctionEnded(address winner, uint amount);
30
31      /// Tao mot hop dong dau gia thong minh voi thoi gian dau gia la
32      /// '_biddingTime' giay, muc gia khoi diem la '_defaultBid' wei
33      /// , vat pham co ma dinh danh '_itemKey'
34      /// nguoi dem vat pham di dau gia co dia chi '_owner'.
35      /// Thoi gian dau gia co the duoc dieu chinh lai.
36      constructor(
37          uint _biddingTime,
38          address _owner,
39          uint _defaultBid,
40          string _itemKey,
41          address addrDatetime
42      ) public {
43          owner = _owner;
44          auctionEnd = now + _biddingTime;
45          highestBid = _defaultBid;
46          defaultBid = _defaultBid;
47          itemKey[owner] = _itemKey;
48          dt = DateTime(addrDatetime);
49      }
50
51      /// Thay doi tinh trang hop dong boi nguoi dem vat pham di dau gia
52      modifier onlyBy(address account) {
53          require(
54              msg.sender == account,
55              "Khong co quyen thuc hien."
56          );
57          _;
58      }
59
60      /// Huy phien dau gia hien tai!
61      function cancel() public onlyBy(owner){
62          ended = true;
63          // Tra tien lai cho nguoi dau gia
64          if (highestBid != defaultBid) {
65              // De nguoi choi tu rut tien
66              pendingReturns[highestBidder] += highestBid;
67              delete highestBidder;
68              highestBid = defaultBid;
69          }
70      }
71
72      /// Kich hoat lai phien dau gia!
73      function activeAgain() public onlyBy(owner) {
74          // Chi duoc kich hoat lai khi con thoi gian dau gia.
75          require(now <= auctionEnd, "Khong the kich hoat lai.");
76          ended = false;
77      }
78
```

```
79      /// Dat thoi gian ket thuc dau gia neu muon thoi gian dau gia keo dai
        hon.
80      function alterAuctionEndTime (
81          uint16 year,
82          uint8 month,
83          uint8 day,
84          uint8 hour,
85          uint8 minute,
86          uint8 second
87      ) public
88        onlyBy(owner) {
89          require(
90              !ended,
91              "Phien da ket thuc! Khong the thay doi!"
92          );
93          uint timeStamp =
94              dt.toTimestamp(year, month, day, hour, minute, second);
95        require(
96            timeStamp > auctionEnd,
97            "Thoi han moi khong lon hon thoi han ban dau!"
98        );
99        auctionEnd = timeStamp;
100     }
101
102     /// Dau gia voi so tien di kem giao dich nay.
103     /// Tien chi duoc hoan lai neu ban khong thang.
104     function bid() public payable {
105
106         // Huy giao dich neu qua han.
107         require(
108             now <= auctionEnd,
109             "Phien dau gia ket thuc!"
110         );
111
112         // Phien dau gia phai dang trong qua tr nh hoat dong
113         require(
114             !ended,
115             "Vat pham da bi huy ban dau gia!"
116         );
117
118         // Khong chap nhan muc gia thap hon muc gia dau hien tai.
119         require(
120             msg.value > highestBid,
121             "Da co gia dau cao hon!!"
122         );
123
124         // Nguoi choi phai dam bao co du tien.
125         require(
126             msg.value <= msg.sender.balance,
127             "Ban khong du tien de tham gia!"
128         );
129
130         if (highestBid != defaultBid) {
```

```
131             // Nguoi choi co gia dau thap hon co the tu rut tien lai
132             pendingReturns[highestBidder] += highestBid;
133         }
134         highestBidder = msg.sender;
135         highestBid = msg.value;
136         emit HighestBidIncreased(msg.sender, msg.value);
137     }
138 
139     /// Rut tien trong truong hop co nguoi khac tra cao hon.
140     function withdraw() public returns (bool) {
141         uint amount = pendingReturns[msg.sender];
142         if (amount > 0) {
143             // Dat lai ve 0 de tranh truong hop gian lan rut nhieu lan.
144             pendingReturns[msg.sender] = 0;
145 
146             // Neu rut tien khong thanh cong
147             if (!msg.sender.send(amount)) {
148                 // Xem nhu chua rut.
149                 pendingReturns[msg.sender] = amount;
150                 return false;
151             }
152         }
153         return true;
154     }
155 
156     /// Ket thuc phien dau gia, gui tien cho nguoi  em  vat  p h m  i
      u   gi  ,
157     /// gui ma vat pham cho nguoi thang cuoc.
158     function auctionEnd() public {
159         // 1. Conditions
160         require(
161             now >= auctionEnd,
162             "Phien dau gia chua ket thuc."
163         );
164         require(
165             !ended,
166             "Phien dau gia khong hoat dong."
167         );
168 
169         // 2. Effects
170         ended = true;
171         emit AuctionEnded(highestBidder, highestBid);
172 
173         // 3. Interaction
174         // Neu co nguoi dau gia
175   if (highestBid != defaultBid) {
176       owner.transfer(highestBid);
177       itemKey[highestBidder] = itemKey[owner];
178       itemKey[owner] = "";
179   }
180   // else do nothing
181   // Thuc te thi he thong se tra lai vat pham cho nguoi dem no di dau gia.
182     }
```

```
}
```

File "DateTime.sol" sử dụng lại từ code mẫu trong đề bài tập lớn.
DateTime.sol

```
pragma solidity ^0.4.24;

contract DateTime {

    struct _DateTime {
        uint16 year;
        uint8 month;
        uint8 day;
        uint8 hour;
        uint8 minute;
        uint8 second;
        uint8 weekday;
    }

    uint constant DAY_IN_SECONDS = 86400;
    uint constant YEAR_IN_SECONDS = 31536000;
    uint constant LEAP_YEAR_IN_SECONDS = 31622400;

    uint constant HOUR_IN_SECONDS = 3600;
    uint constant MINUTE_IN_SECONDS = 60;

    uint16 constant ORIGIN_YEAR = 1970;

    function isLeapYear(uint16 year) public pure returns (bool) {
        if (year % 4 != 0) {
            return false;
        }
        if (year % 100 != 0) {
            return true;
        }
        if (year % 400 != 0) {
            return false;
        }
        return true;
    }

    function leapYearsBefore(uint year) public pure returns (uint) {
        year -= 1;
        return year / 4 - year / 100 + year / 400;
    }

    function getDaysInMonth(uint8 month, uint16 year) public pure returns (
    uint8) {
        if (month == 1 || month == 3 || month == 5 || month == 7 || month
    == 8 || month == 10 || month == 12) {
            return 31;
        }
        else if (month == 4 || month == 6 || month == 9 || month == 11) {
            return 30;
```

```
48          }
49          else if (isLeapYear(year)) {
50              return 29;
51          }
52          else {
53              return 28;
54          }
55      }
56
57      function parseTimestamp(uint timestamp) internal pure returns (
        _DateTime dt) {
58          uint secondsAccountedFor = 0;
59          uint buf;
60          uint8 i;
61
62          // Year
63          dt.year = getYear(timestamp);
64          buf = leapYearsBefore(dt.year) - leapYearsBefore(ORIGIN_YEAR);
65
66          secondsAccountedFor += LEAP_YEAR_IN_SECONDS * buf;
67          secondsAccountedFor += YEAR_IN_SECONDS * (dt.year - ORIGIN_YEAR -
        buf);
68
69          // Month
70          uint secondsInMonth;
71          for (i = 1; i <= 12; i ++) {
72              secondsInMonth = DAY_IN_SECONDS * getDaysInMonth(i, dt.year);
73              if (secondsInMonth + secondsAccountedFor > timestamp) {
74                  dt.month = i;
75                  break;
76              }
77              secondsAccountedFor += secondsInMonth;
78          }
79
80          // Day
81          for (i = 1; i <= getDaysInMonth(dt.month, dt.year); i ++) {
82              if (DAY_IN_SECONDS + secondsAccountedFor > timestamp) {
83                  dt.day = i;
84                  break;
85              }
86              secondsAccountedFor += DAY_IN_SECONDS;
87          }
88
89          // Hour
90          dt.hour = getHour(timestamp);
91
92          // Minute
93          dt.minute = getMinute(timestamp);
94
95          // Second
96          dt.second = getSecond(timestamp);
97
98          // Day of week .
```

```
99          dt.weekday = getWeekday(timestamp);
100     }
101
102     function getYear(uint timestamp) public pure returns (uint16) {
103         uint secondsAccountedFor = 0;
104         uint16 year;
105         uint numLeapYears;
106         // Year
107         year = uint16(ORIGIN_YEAR + timestamp / YEAR_IN_SECONDS);
108         numLeapYears = leapYearsBefore(year) - leapYearsBefore(ORIGIN_YEAR)
    ;
109         secondsAccountedFor += LEAP_YEAR_IN_SECONDS * numLeapYears;
110         secondsAccountedFor += YEAR_IN_SECONDS * (year - ORIGIN_YEAR -
    numLeapYears);
111         while (secondsAccountedFor > timestamp) {
112             if (isLeapYear(uint16(year - 1))) {
113                 secondsAccountedFor -= LEAP_YEAR_IN_SECONDS;
114             }
115             else {
116                 secondsAccountedFor -= YEAR_IN_SECONDS;
117             }
118             year -= 1;
119         }
120         return year;
121     }
122
123     function getMonth(uint timestamp) public pure returns (uint8) {
124         return parseTimestamp(timestamp).month;
125     }
126
127     function getDay(uint timestamp) public pure returns (uint8) {
128         return parseTimestamp(timestamp).day;
129     }
130
131     function getHour(uint timestamp) public pure returns (uint8) {
132         return uint8((timestamp / 60 / 60) % 24);
133     }
134
135     function getMinute(uint timestamp) public pure returns (uint8) {
136         return uint8((timestamp / 60) % 60);
137     }
138
139     function getSecond(uint timestamp) public pure returns (uint8) {
140         return uint8(timestamp % 60);
141     }
142
143     function getWeekday(uint timestamp) public pure returns (uint8) {
144         return uint8 ((timestamp / DAY_IN_SECONDS + 4) % 7);
145     }
146
147     function toTimestamp(uint16 year, uint8 month, uint8 day) public pure
    returns (uint timestamp) {
148          return toTimestamp(year, month, day, 0, 0, 0);
```

```
149     }
150
151     function toTimestamp (uint16 year, uint8 month, uint8 day, uint8 hour)
        public pure returns (uint timestamp) {
152         return toTimestamp(year, month, day, hour, 0, 0);
153     }
154
155     function toTimestamp(uint16 year, uint8 month, uint8 day, uint8 hour,
        uint8 minute) public pure returns (uint timestamp) {
156         return toTimestamp(year, month, day, hour, minute, 0);
157     }
158
159     function toTimestamp(uint16 year, uint8 month, uint8 day, uint8 hour,
        uint8 minute, uint8 second) public pure returns (uint timestamp) {
160         uint16 i;
161
162         // Year
163         for (i = ORIGIN_YEAR; i < year; i ++) {
164             if (isLeapYear(i)) {
165                 timestamp += LEAP_YEAR_IN_SECONDS;
166             }
167             else {
168                 timestamp += YEAR_IN_SECONDS;
169             }
170         }
171         // Month
172         uint8[12] memory monthDayCounts;
173         monthDayCounts[0] = 31;
174         if (isLeapYear(year)) {
175             monthDayCounts[1] = 29;
176         }
177         else {
178             monthDayCounts[1] = 28;
179         }
180         monthDayCounts[2] = 31;
181         monthDayCounts[3] = 30;
182         monthDayCounts[4] = 31;
183         monthDayCounts[5] = 30;
184         monthDayCounts[6] = 31;
185         monthDayCounts[7] = 31;
186         monthDayCounts[8] = 30;
187         monthDayCounts[9] = 31;
188         monthDayCounts[10] = 30;
189         monthDayCounts[11] = 31;
190         for (i = 1; i < month ; i ++) {
191             timestamp += DAY_IN_SECONDS * monthDayCounts[i - 1];
192         }
193         // Day
194         timestamp += DAY_IN_SECONDS * (day - 1);
195         // Hour
196         timestamp += HOUR_IN_SECONDS * (hour);
197         // Minute
198         timestamp += MINUTE_IN_SECONDS * (minute);
```

```
199             // Second
200             timestamp += second;
201             return timestamp;
202         }
203 }
```

Tài liệu

[Alex93] Vladimir Alexiev. *Applications of Linear Logic to Computation: An Overview*, 1993.

[Ser18] Sergy Nosikov. *What are smart contracts?* https://www.cryptoninjas.net/what-are-smart-contracts/. Truy cập vào 01/6/2018.

[Sza96] Nick Szabo. *Smart Contracts: Building Blocks for Digital Markets.*. Trên Extropy #16, 1996.

[1] *Smart Contracts: The Blockchain Technology That Will Replace Lawyers.* https://blockgeeks.com/guides/smart-contracts/. Truy cập vào 01/6/2018.

[2] *Smart Contracts: 12 Use Cases For Business And Beyond.* https://www.ccn.com/smart-contracts-12-use-cases-for-business-and-beyond/. Truy cập vào 01/6/2018.

[3] *5 Applications Of Smart Contracts.* https://disruptionhub.com/smart-contract-uses/. Truy cập vào 01/6/2018.

[4] *Computational Aspects of Linear Logic.* Nhà xuất bản MIT thuộc Viện Công nghệ Massachusetts (1996), trang 2.