ISSN 2735-4822 (Online) \ ISSN 2735-4814 (print)

Hoare logic and its role in computer programs

Master. Heba Mamdoh Mukhtar Mohamed
Master's researcher-Department of Philosophy - Faculty of Women for Arts, Science & Education, Ain Shams University.
heba.mamdoh@women.asu.edu.eg
Prof. Siham Mahmoud Al Nowaihi
Professor of Logic and Philosophy of Science- Faculty of Women for Arts, Science & Edu Ain Shams University.
seham.elnewhy@women.asu.edu.eg
Assoc.Prof. Maysa Abdou Ali El Sayed
Associate Professor of Logic- Faculty of Women for Arts, Science & Edu, Ain Shams University
maysa.ali@women.asu.edu.eg

Receive Date: 6 October 2023, Revise Date: 31 October 2023
Accept Date: 2 November 2023.
DOI: 10.21608/BUHUTH.2023.241092.1576
Volume 4 Issue 5 (2024) Pp.164-197.

Abstract

Designing and developing high-quality systems that meet their requirements is extremely important. Especially with the ever-increasing complexity of computer systems, failure of the system in its mission or safety can cause many problems such as: cost overruns, loss of life or severe economic losses; Therefore, many scientists have directed their attention to detecting errors in these programs and treating them, based on sound logical foundations. Hoare's logic is considered the first logic that presented a formal format for application in the field of formal verification of the verification of the program. Therefore, this research aims to clarify and explain the role of Hoare's logic in verifying the program. By identifying the reasons for its appearance, the stages of its development, its concept, presenting and analyzing the components of its formal format, and explaining how to apply it in verifying the validity of the program. The researcher followed the historical approach and the comparative critical analytical approach. It becomes clear to us that logic is not separated from reality, but rather keeps pace with it considering scientific and technological progress. In addition, computer science students cannot do without studying logic; It provides them with the basis for understanding how computers work, and how to deal with them. If the programmer wants to avoid errors in the code and verify the validity of his program, he must apply the foundations and rules of Hoare's logic. Therefore, Hoare's logic gained the attention of logicians and computer scientists, and many contributions appeared that sought to develop it considering the continuous scientific progress. To verify and prove the validity of programs of all kinds, and even develop them; Therefore, the researcher recommends directing more attention in the Arab world to scientific research on new types of logic and their applications in the fields of computer science.

***Keywords*:** Hoare logic, Hoare triple, Formal verification, assertions, Specification, program verification

منطق هور ودَوْره في برامج الحاسوب

هبه ممدوح مختار محمد
باحثة ماجستير- قسم الفلسفة
كلية البنات، جامعة عين شمس، مصر
heba.mamdoh@women.asu.edu.eg

أ.د/سهام محمود النويهي
أستاذ المنطق وفلسفة العلوم
كلية البنات، جامعة عين شمس، مصر
seham.elnewhy@women.asu.edu.eg

ا.م.د/ مـايسة عبده علي السيد
أستاذ المنطق المساعد
كلية البنات، جامعة عين شمس، مصر
maysa.ali@women.asu.edu.eg

المستخلص

يُعد تصميم وتطوير أنظمة عالية الجودة تلبي متطلباتها أمرًا في غاية الأهمية؛ خاصة مع التعقيد المتزايد باستمرار لأنظمة الحاسوب، فمن الممكن أن يتسبَّب فشل النظام في مهمته أو سلامته إلى مشكلات عديدة؛ مثل: تجاوز التكلفة، وفقدان في الأرواح أو خسائر اقتصادية وخيمة؛ لذلك وجَّه العديد من العلماء اهتمامهم للكشف عن أخطاء تلك البرامج ومعالجتها، استنادًا إلى أسس منطقية سليمة، ويُعتبر منطق هور أول منطق قدَّم نسقًا صوريًّا لتطبيقه في مجال التحقُّق الصوري من صحة البرنامج؛ لذا يهدف هذا البحث إلى توضيح وبيان دور منطق هور في ذلك؛ من خلال الوقوف على أسباب ظهوره، ومراحل تطوره، ومفهومه، وعرض وتحليل مكونات نسقه الصوري، وبيان كيفية تطبيقه، وقد اتبعت الباحثة في ذلك المنهج التاريخي، والمنهج التحليلي النقدي المقارَن. ليتبيَّن لنا أن المنطق لا ينفصل عن الواقع، بل يسايره في ظل التقدم العلمي والتكنولوجي. بالإضافة إلى أن دارسي علوم الحاسوب لا يمكنهم الاستغناء عن دراسة المنطق؛ فهو يوفِّر لهم الأساس لفهم كيفية عمل أجهزة الحاسوب، وكيفية التعامل معها، فإذا أراد المبرمج أن يتجنَّب الأخطاء في الكود البرمجي، ينبغي أن يتحقَّق من صحة برنامجه، باستخدام أسس وقواعد منطق هور؛ لذا حاز منطق هور على اهتمام علماء المنطق، وعلماء الحاسوب، فظهرت العديد من الإسهامات التي سعت لتطويره في ظل التقدم العلمي المستمر؛ من أجل التحقُّق وإثبات صحة البرامج بكافة أنواعها، بل وتطويرها أيضًا؛ لذلك تُوصي الباحثة بتوجيه المزيد من الاهتمام في عالمنا العربي بالأبحاث العلمية، حول الأنواع الجديدة للمنطق وتطبيقاتها في مجالات علوم الحاسوب.

الكلمات المفتاحية: منطق هور، ثلاثية هور، التحقُّق الصوري، التقريرات، المواصفات، التحقُّق من البرنامج.

المقدمة

مرَّ المنطق بالعديد من التطورات؛ فمنذ ظهور المنطق الرياضي، توالى في الظهور أنواع عديدة من المنطق ترتكز على عدة محاور: **المحور الأول:** الانطلاق من المنطق الرياضي باعتباره الأساس الأول لجميع أنواع المنطق، **المحور الثاني:** يعتبر كل منطق تطورًا للمنطق السابق عليه، **المحور الثالث:** كل نوع من الأنواع له نسق يحكمه ويُميِّزه عما عداه. **المحور الرابع:** كل منطق له محوران: نظري، وتطبيقي. **المحور الخامس:** بعض أنواع المنطق تخدم علوم الحاسوب Computer Science بصفة عامة، والذكاء الاصطناعي Artificial intelligence بصفة خاصة، لكن الاختلاف يكمن في كيفية الخدمة تبعًا لروح ومشكلات كل عصر.

تعريف وتحديد مشكلة البحث

من إحدى المشكلات التي كثَّف علماء المنطق وعلماء الحاسوب جهودهم لإيجاد الحلول لها، هي الأخطاء البرمجية؛ فهي قديمة منذ ظهور مجال البرمجة، فلا يخلو كود برمجي من الأخطاء التي تؤثر على سلوك البرنامج عند تشغيله فيؤدي إلى فشله. ولكن في ظل التقدم العلمي والتكنولوجي أصبحت الأخطاء أكثر ضررًا، خاصة في ظل التعقيد المتزايد في أنظمة الحاسوب، فكل ما حولنا أصبح عبارة عن آلة مبرمَجة، بداية من الهاتف المحمول بكافة تطبيقاته، حتى الأجهزة المنزلية، والطبية، العسكرية... إلخ. فهل يمكن أن أثق في أن جهاز تنظيم ضربات القلب لن ينهار فجأة؟ هل يمكن لجهاز الميكروويف الوصول إلى محفظتي؟ فعندما يكتب المبرمج الأكواد لمثل هذه البرامج، هل يتبع طريقة منطقية لضمان صحتها عند التشغيل، وأن يضمن سلامتها بما لا تُلحِق ضررًا بمستخدميها؟

ولّدت هذه الأسئلة ودفعت بخطي متزايدة نحو تطوير مجال التحقُّق من البرامج، فظهرت العديد من الإسهامات التي شارك فيها علماء المنطق، وعلماء الحاسوب، لوضع أسس منطق جديد يُسمَّى "منطق هور"، للكشف عن الأخطاء البرمجية، وضمان سلامتها بواسطة وضع نسق صوري، أهم ما يميزه هي "ثلاثية هور"، التي من خلالها يمكن الكشف عن صحة البرنامج من خلال تتبُّع سلوكه قبل وبعد تنفيذ البرنامج، ولم يكتف "هور" بذلك بل حاول البرهنة على صحة البرنامج لاستنتاج سلوكياته؛ من خلال وضع مجموعة من البديهيات وقواعد الاستدلال التي قدم تعريفات لها باستخدام ثلاثية، من هنا جاء البحث بعنوان " **منطق هور ودَوْره في برامج الحاسوب"**

تساؤلات البحث

استنادًا إلى العرض السابق لطبيعة مشكلة البحث من حيث متغيراتها التابعة والمستقلة، سعت الباحثة للإجابة عن عدة تساؤلات؛ هي:

1- ما العوامل التي ساهمت في ظهور منطق هور؟ وما مراحل ظهوره؟
2- ما الذي يُميِّز المبرمج الجيد عن المبرمج غير الجيد؟
3- ماذا يُقصد بمنطق هور؟
4- ماذا يُقصد بثلاثية هور؟ وممَّ تتكون؟ وكيف نحكم عليها بالصحة وعدم الصحة؟
5- كيف تساهم المواصفات في معرفة سلوك البرنامج؟
6- كيف يتأكد المبرمج من أن البرنامج ينفِّذ الغرض المقصود منه؟

7- ماذا يُقصد بالتقريرات؟ وممَّ تتكون؟ وما أنواعها؟
8- ما الفرق بين اختبار البرنامج، والتحقُّق من البرنامج، وإثبات صحة البرنامج؟
9- ممَّ تتكون قواعد النسق الاستنباطي في منطق هور؟ وإلى ماذا تهدف؟

أهداف البحث

يهدف البحث إلى إلقاء الضوء على الجوانب التالية:

1- التأكيد على أهمية المنطق وأنه لا ينفصل عن الواقع؛ فهو ذو منحًى تطبيقي في مجالات علوم الحاسوب.
2- بيان دور منطق هور في التحقُّق من صحة البرامج.
3- بيان المعايير الأساسية للبرنامج الصحيح.
4- توضيح العلاقة بين منطق هور وبين منطق الدرجة الأولى.
5- توضيح الفرق بين اختبار البرنامج، والتحقُّق من صحة البرنامج، وإثبات صحته.
6- توضيح الفرق بين متغيرات البرنامج، والمتغيرات المنطقية.
7- توضيح العلاقة بين الشروط المسبقة والشروط اللاحقة وأمر البرنامج.
8- بيان دور البديهيات وقواعد الاستدلال في إثبات صحة البرنامج.
9- فتح مجالات جديدة للبحث العلمي حول استخدام المنطق في إثبات صحة وسلامة البرامج.

أهمية البحث

تنبع أهمية البحث من:

1- أهمية صقل المبرمج بمهارة تُمكّنه من الكشف عن أخطاء الأكواد البرمجية؛ وذلك من خلال استخدام التحقُّق من البرامج لضمان صحتها وسلامتها في التنفيذ والاستخدام. خاصة مع برامج السلامة الحرجة التي قد يُشكل فشلها خطرًا على حياة الإنسان، أو خسائر اقتصادية وخيمة، أو أضرارًا بيئية، وما إلى ذلك.
2- أهمية ومكانة منطق هور باعتباره أول منطق يُطبق في مجال التحقُّق من صحة البرامج؛ لذلك استند العلماء إليه لإنشاء أنواع أخرى من المنطق تهدف إلى التحقُّق من صحة البرامج.

وتم اختيار الموضوع انطلاقًا من عدة دوافع وأسباب، يمكن إجمالها فيما يلي:

- **مبررات ذاتية:**

ارتباط الموضوع باهتمام الباحثة الذي يتمثَّل في اختيار موضوع، يُبرز الجانب التطبيقي للمنطق في مجال علوم الحاسوب.

- **مبررات موضوعية:**

1- تتمثَّل المبررات الموضوعية في عدم وجود كتابات عربية تناولت هذا الموضوع.

منهج البحث

اعتمدت الباحثة في هذا البحث على عدة مناهج:

- **المنهج التاريخي:** المُستخدم في تتبُّع مراحل ظهور منطق هور وتطوره.
- **المنهج التحليلي المقارن:** يتمثَّل في تحليل أفكار "هور"، وتحليل بعض الأكواد والرموز المنطقية، وبيان أوجه الشبه والاختلاف بين بعض المصطلحات.
- **المنهج النقدي:** في التقييم والتعليق على منطق هور.

محتويات البحث

قسَّمتُ البحث إلى مقدمة، وأربعة عناصر رئيسية، والأبحاث المستقبلة حول الموضوع، وخاتمة وتوصيات، وذلك كالآتي:

المقدمة: بيَّنتُ فيها إشكالية الموضوع، وأهدافه، وأهميته، وأسباب اختياره والمنهج المستخدَم، وخطة البحث.

العنصر الأول: (**نشأة وتطور منطق هور**) خصَّصته للحديث عن أسباب ظهور منطق هور، وتتبُّع وعرض مراحل نشأته وتطوره، وتوضيح مفهومه.

العنصر الثاني:(**ثلاثية هور**) خصَّصته لشرح وبيان البناء التركيبي والدلالي لمنطق هور.

العنصر الثالث: (**قواعد النسق الاستنباطي**) خصَّصته لشرح وتوضيح قواعد النسق لمنطق هور بما يتضمن من بديهيات، وقواعد الاستدلال.

العنصر الرابع: (نقد وتقييم) تناولت فيه نقد العلماء وتقيمهم لمنطق هور، وجهودهم نحو تطويره.

الخاتمة والتوصيات: ضمَّنتها أهم النتائج التي تم التوصَّل إليها، والتوصيات المُقدَّمة من الباحثة.

******

أولًا: نشأة وتطور منطق هور

1- عوامل الظهور.

أ- التحقُّق الصوري: Formal verification

إن البرامج لا تخلو من الأخطاء فهذه طبيعتها منذ نشأة البرمجة، وهذا ما أكَّد عليه مهندسو علوم الحاسوب، بقولهم: **"تُعد الأخطاء جزءًا متأصلًا في البرمجيات منذ بداية تخصص البرمجة. حيث تميل إلى الاختباء في أماكن غير متوقَّعة، وعندما يتم تشغيلها، يمكن أن تُسبِّب أضرارًا كبيرة** (Harmim, Marcin,& Pavela, 2019. p.1) فمهما حرص المبرمجون على كتابة أكواد دقيقة، فإنها قد لا تخلو من الأخطاء التي لن تُكتشف إلا بعد تشغيل البرنامج، وحينها ستؤدي إلى أضرار متفاوتة في شدتها، ومتباينة في طبيعتها تبعًا لنوع البرنامج.

لذا ركَّز العلماء بشكل متزايد على الأكواد البرمجية code الخالية من الأخطاء (أو خالية من الأخطاء قدر الإمكان)، فإحدى الأساليب التقليدية هي الاختبارtesting الذي يهدف إلى فحص سلوك البرنامج من خلال تنفيذه على فئة معينة من البيانات؛ من أجل اكتشاف الأخطاء وإزالتها، فكلما قل مقدار الخطأ الذي تم اكتشافه وإزالته بنجاح، كان البرنامج أكثر صحة (Adrion, Branstad & Cherniavsky,1982,p.161; Osho, ,Francisca,2013.p.68)؛ مع ذلك، لم يعد هذا الأسلوب التقليدي مناسبًا مع تعقيدات البرنامج **"فإذا كان الاختبار وسيلة فعَّالة للغاية لإظهار وجود الأخطاء، لكنه غير مناسب لإظهار غيابها"**(Dijkstra, 1972,p.864).

من ثمَّ لا يمكن للمبرمج أن يثق في البرنامج إلا إذا تحقق من صحته من خلال تقديم دليل مقنِع، ولكن لا ينبغي له أن يصنع البرنامج أولًا ثم يثبت صحته، على العكس من ذلك يجب على المبرمج أن يترك الدليل والبرنامج ينموان جنبًا إلى جنب (Dijkstra, 1972,p.864). **فكيف يمكن إذًا تقديم دليل على صحة البرنامج؟**

يُقصد بالتحقق الصوري من البرنامج تقديم دليل مقنع بناءً على نظرية سليمة أن سلوك البرنامج يفي بالمواصفات المحددة (Naumann, 2009, p.1.). أي يهدف إلى تتبُّع سلوك البرنامج للتأكُّد من استيفائه بالمواصفات التي حدَّدها المبرمِج وفقًا لقواعد منطقية وأسس سليمة.

بذلك يمكن أن يؤدي التحقُّق من البرنامج إلى زيادة موثوقية البرنامج بطرق قد يتم التغاضي عنها في الاختبارات التقليدية. يأتي الاختبار التقليدي بشكل جوهري في نهاية تصميم البرنامج، في حين قد يتم تقديم التحقُّق من البرنامج في البداية، مما يؤدي إلى إزالة الأخطاء في وقت أبكر مما كان ممكنًا، وهذا بدوره يمكن أن يجعل تطوير البرنامج سريعًا(Kurshan, 2000,p.545.).

ب- تنمية مهارات المبرمِج الجيد:

تُعتبر دراسة المنطق أساسًا مهمًّا لا بد أن يبدأ به دارسو علوم الحاسوب، والإعراض عنها لن يؤسس مبرمج حاسوب جيدًا في مجاله، وقد أكَّد بعض علماء الحاسوب على أهمية دراسة وفهم المنطق للطلاب؛ إذ يقولون: **"يُعد فهم المنطق وتخطيط الخوارزمية قبل البدء في كتابة الكود أساسًا مهمًّا للغاية، ولكن أحد العادات السيئة للطلبة هو تخطِّي هذه المرحلة المهمة"**(Leroy & Braunchweig,2018,P.11). علاوة على ذلك تُعد دراسة المنطق أكثر ملاءمة لطلاب هندسة وعلوم الحاسوب؛ فالمنطق مهم بشكل خاص لأنه الأساس الرياضي للبرمجة، فهو يُستخدَم لإضفاء الطابع الصوري على دلالات لغات البرمجة ومواصفات البرامج specification of programs، والتحقُّق من صحة البرامج verify the correctness of programs (Ben-Ari,2001, P(Preface)).

لذا ميَّز العلماء بين نوعين من المبرمجين، المبرمج الجيد وغير الجيد. **فالمبرمج غير الجيد** هو من يكتب برامجه من خلال **"التجربة والخطأ" Trial and error** (Schreiner,2021, p (xxiv).). حيث يقوم بكتابة الأكواد البرمجية ثم يقوم بتشغيل البرنامج؛ ليكشف عن سلوك البرنامج فإذا كان هناك خطأ ما يقوم بتعديله، وهكذا حتى ينتهي من كتابة البرنامج بشكل صحيح(Schreiner,2021, p (xxiv)) .

فقد صرح عالم الحاسوب البريطاني تشارلز أنتوني ريتشارد هور Charles Antony Richard Hoare (*) برفضه استخدام المبرمج لهذا الأسلوب؛ لأنه يستغرق وقتًا كبيرًا، وتكلفة عالية، ولا يحقق نتائج دقيقة، فضلًا عن الأخطاء التي ستظل كامنة داخل البرنامج دون اكتشافه، حيث يقول: **" الطريقة التي يستخدمها المبرمج في الوقت الحاضر لإقناع نفسه بصحة البرنامج هو التجربة في حالات معينة، فإذا كانت النتائج التي توصل إليها لا تتفق مع هدف البرنامج، فإنه يقوم بتعديلها. غالباً ما يستغرق هذا أكثر من نصف الوقت المستغرق في البرمجة بأكملها بالإضافة إلي ارتفاع التكلفة لتصحيح هذه الأخطاء "** (Hoare, 1969, p.579).

أما **المبرمج الجيد** فهو من يطبِّق الاستنتاج العقلاني rational reasoning؛ لمعرفة سلوك برنامجه من نصوص البرنامج وحده. فقبل أن يطلق برنامجه، لا بد أن يختبره بخطوات منطقية، والهدف من هذا الاختبار هو التأكد من استيفاء البرنامج لمجموعة من الخصائص الأساسية Osho **.**(&Francisca,2013.p.67)

1- **الموثوقيةreliability** : التي هي مقياس لصحة نتيجة البرنامج.
2- **سهولة الاستخدامusability** : والتي توضح سهولة استخدام البرنامج للغرض المقصود منه.
3- **قابلية النقلportability** : قدرة البرنامج على العمل على منصات مختلفة.
4- **قابلية الصيانةmaintainability** : إمكانية تعديل أو ترقية أو تحسين قدرة البرنامج.

(*)= تشارلز أنتوني ريتشارد هورCharles Antony Richard Hoare (1934-....) هو عالم حاسوب بريطاني بكولومبو Colombo في سريلانكا Sri Lanka حصل على درجة البكالوريوس في الكلاسيكيات من جامعة أكسفورد عام 1956. بعد التخرُّج أخذ بعض الدورات على مستوى الدراسات العليا في الإحصاء وبرمجة الحاسوب في أكسفورد. وفي عام 1959 انتقل هور إلى روسيا للعمل في الدراسات العليا في نظرية الاحتمالات والترجمة الحاسوبية للغات البشرية في جامعة موسكو الحكومية. له العديد من الأعمال المنشورة التي كان مهتمًّا فيها بالمنطق وعلوم الحاسوب؛ وأهمهم: "**أسس البديهيات لبرمجة الحاسوب** An Programming Axiomatic Basis for Computer **(1969 م)** التي قدم فيها أسس منطقه الجديد. وله أوراق بحثية أخرى؛ مثل: عمليات التواصل المتسلسل Communicating Sequential Processes (1985م)؛ المنطق الرياضي ولغات البرمجة Mathematical Logic and Programming Languages (1985م).مقالات في علوم الحوسبةComputing Science **(1989**م)، والتطورات في التزامن والتواصل (1991م) Developments in Concurrency and Communication. ومن أهم إنجازاته: ابتكر Quicksort وهي خوارزمية حاسوبية للبحث الفعَّال عن المعلومات في جداول الحاسوب. حاز هور على العديد من الجوائز تقديرًا لإنجازاته العلمية؛ مثل: جائزة تورين (1980م)، أعلى وسام شرف في علوم الحاسوب لمساهماته الأساسية في تعريف وتصميم لغات البرمجة. وميدالية فاراداي Faraday Medal (1985م)، وجائزة كيوتو = Kyoto Prize (2000م)، وميدالية جون فون نيومان من معهد مهندسي الكهرباء والإلكترونيات (2011م). حصل على لقب فارس من قِبل إليزابيث الثانية في عام 2000م لمساهماته في علوم الحاسوب. **انظر**.(Hosch, 2009; Jones, 2010, p.101).

5- **المتانةrobustness** : قدرة البرنامج على تحمُّل الظروف غير العادية.

على سبيل المثال لا الحصر ترتبط الموثوقية ارتباطًا وثيقًا بصحة الخوارزمية ومقدار الأخطاء الموجودة في أكواد البرنامج (Osho &Francisca,2013.p.67)، وأضاف هور أن الموثوقية تُعَد من المشكلات التي يسعى إليها العلماء لإيجاد الحلول لها؛ إذ يقول:

"يبدو أن ممارسة إثبات البرامج تؤدي إلى حل ثلاثة من أكثر المشكلات إلحاحًا في البرمجيات؛ وهي: الموثوقية، والتوثيق documentation، والتوافق compatibility"(Hoare, 1969, p.580).

من أجل ضمان حل تلك المشكلات أكَّد العلماء على أهمية إضفاء الطابع الصوري Formal method على عملية التفكير هذه في شكل نظرية منطقية logical theory، والتنبؤ بسلوك البرنامج عن طريق الاستنباط المنطقي "logical deduction"(Schreiner,2021, p (xxiv)).

يمكن القول إذًا بأن التحقق الصوري بمثابة إجابة عن تساؤل "هل تم بناء البرامج بشكل صحيح؟ **ولكن أي نوع من البرامج التي ينبغي التحقُّق من صحتها؟**

ج- برامج السلامة الحرجة: safety-critical system

يُستخدم التحقق الصوري بشكل خاص مع برامج السلامة الحرجة؛ مثل: أجهزة التحكم في المفاعلات النووية، أنظمة فرملة السيارة، والطائرات التي تعمل بالأسلاك، والمعدَّات الطبية التي قد يشكِّل فشلها خطرًا على حياة الإنسان، ويؤدي إلى خسارة اقتصادية كبيرة، أو تُسبِّب أضرارًا بيئية جسيمة. لا تُعتبر السلامة فقط بالنسبة إلى البرامج، ولكن أيضًا للأجهزة وللمستخدمين. حيث إن فشل النظام يمكن أن يؤدي إلى عواقب كثيرة.(Amarendra, & Rao, 2011,p.37; Feng & Müller, 2018,p.70;)

أ- أمثلة:

1- خطأ ثيراك-25، الأشعة السينية (Therac-25, X-ray) (Rose, 1994.)

- **محتوى الخطأ**: جرعات زائدة الإشعاع بواسطة آلة علاج السرطان ثيراك 25 التي بها خلل برمجي نتج عنها 5 وفيات في عام 1985.
- **السبب**: حالة السباق بين مهام البرنامج في الذاكرة.

2- صاروخ باتريوت Patriot Missile في المملكة العربية السعودية

محتوى الخطأ: فشل الصاروخ في أثناء حرب الخليج، فأدى إلى مقتل 28 شخصًا في عام 1991.

السبب: مشكلة برمجية أدت إلى فشل النظام بسبب أخطاء حسابية، أدت إلى حساب غير دقيق للوقت منذ بدء التشغيل (Feng& Müller, 2018,p.70).

3- آريان 5 (Ariane 5) (Manna & Waldinger., 1978,p.200)

- **محتوى الخطأ**: خرج أول مسبار فضائي أمريكي إلى كوكب الزهرة عن مساره في عام 1996 وكان لا بد من تدميره بعد 37 ثانية من الإقلاع ونتج عنه خسارة بحوالي 370 مليون دولار أمريكي.
- **السبب**: خطأ في أحد برامج التوجيه في جهاز الحاسوب الموجود على متنه.

تلك بعض الأمثلة التي توضِّح أن مهام أنظمة السلامة الحرجة يمكن أن تكون برامجها غير آمنة. لكن ليس كل أخطاء البرنامج لديها مثل هذه العواقب؛ فالمبرمج الذي يفهم التحقُّق سوف يتجنَّب الكثير من الأخطاء؛ لذلك يمثِّل التحقُّق الصوري دورًا مهمًّا للتأكُّد من أن سلوك البرنامج يتفق مع هدفه، مما يساهم في إنقاذ الأرواح أو منع ملايين الأضرار(Swenker, 2012,p.1).

يُعتبر منطق هور أول نسق منطقي استُخدم لفترة طويلة للتحقُّق من البرنامج. (Manna & Waldinger., 1978,p.200) وقد مرَّ بعدة مراحل ساهمت في نشأته وتطوره؛ من خلال إسهام العديد من المناطقة وعلماء الحاسوب.

2- النشأة.

يُعد الاهتمام بشأن صحة برامج الحاسوب قديمًا قدم أجهزة الحاسوب نفسها، ولكن سنركز هنا على إسهامين سبقا ظهور منطق هور، وهما إسهام كلٍّ من: عالم الحاسوب والمنطق الأمريكي آلان تورينAlan Turing (*) (1912- 1954)، وعالم الحاسوب الأمريكي "روبرت فلويد .Robert W Floyd (**)(Apt, & Olderog, 2019,p.753)

(*) **آلان ماتيسون تورين (1912 –1954)** هو عالم الرياضيات والمنطق البريطاني وُلِد بلندن في إنجلترا. قدم إسهامات بارزة في: الرياضيات، وتحليل الشفراتcryptanalysis والمنطق، والفلسفة وكذلك في المجالات الجديدة التي سُمِّيت لاحقًا بعلوم الحاسوب والعلوم المعرفية cognitive science والذكاء الاصطناعي والحياة الاصطناعيةartificial life . التحق بجامعة كامبريدج لدراسة الرياضيات في عام 1931، تم انتخابه بعد تخرجه في عام 1934 للزمالة في كلية كينجز College000–000–King's تقديرًا لأبحاثه في نظرية الاحتمالات. حصل على درجة الدكتوراه في المنطق الرياضي عام 1938 من جامعة برينستون Princeton University ومن أبرز إسهاماته: يُعد الأب المؤسس للذكاء الاصطناعي والعلوم المعرفية الحديثة، وكان من أوائل روَّاد الدفاع عن الفرضية القائلة بأن الدماغ البشري هو في جزء كبير منه آلة حوسبة رقمية. لقد افترض أن القشرة الدماغية عند الولادة هي "آلة غير منظَّمة" والتي من خلال "التدريب" تصبح منظمة "في آلة عالمية أو شيء من هذا القبيل". اقترح تورين ما أصبح يُعرف لاحقًا باسم اختبار تورين كمعيار لما إذا كان الحاسوب الاصطناعي يفكِّر (1950). **انظر** (Hodges,2014).

(**) "**روبرت فلويد**" (1936– 2001) هو عالم حاسوب أمريكي وُلِد بنيويورك في الولايات المتحدة. حصل على درجة البكالوريوس عام 1953م في السابعة عشر من عمره، ولكن طموحه وموهبته جعلاه يتجه إلى دراسة علوم الحاسوب فعكف على تعلُّم البرمجة فأصبح مبرمجًا ومُحلِّلًا. في عام 1958م حصل على درجة البكالوريوس في الفيزياء من جامعة شيكاغو. انضم فلويد إلى هيئة التدريس في معهد كارنيجي للتكنولوجيا بصفته أستاذًا مساعدًا. أصبح عضوًا في هيئة تحرير أهم مجلة تقنية في علوم الحاسوب في ذلك الوقت "مجلة جمعية آلات الحوسبة" Journal of the Association of Computing Machinery. انتقل إلى ستانفورد وأصبح أستاذًا في عام 1970م. وكان رئيسًا لقسم علوم الحاسوب في الفترة من 1973–1976م . قدم فلويد أول ورقة بحثية بعنوان "**لغة وصفية لمعالجة الرموز**"**A descriptive language for symbol manipulation** " (1961م) تضمنت تدوينًا جديدًا لوصف ما يجري بالفعل في عملية ترجمة برنامج من لغة برمجة إلى أخرى. حصل على جائزة تورين عام 1986م – أعلى وسام في علوم الحاسوب– وذلك للمساعدة في إيجاد الحقول الفرعية التالية المهمة في علوم الحاسوب: دلالات لغات البرمجة، التحقُّق التلقائي من البرنامج، توليف البرامج تلقائيًّا synthesis automatic program، تحليل الخوارزميات. وقد ساهم بشكل كبير في فهم معنى البرامج وكان له تأثير كبير على "أنتوني هور" **انظر** (Knuth, December 2003; Lee, 1995)

أ- **إسهام آلان تورين:**

قدم "آلان تورين" عام 1949م "عرضًا تقديميًّا بعنوان: **"التحقُّق من نمط(*) كبير"** Checking a Large Routine...بدأ تورينج بالسؤال **"كيف يمكن للمرء أن يفحص البرنامج بمعنى التأكد من صحته؟"** (Turing,1949,p.67.).

كي لا تكون تلك المهمة صعبة اقترح تورين أنه **"على المبرمج أن يقدم عددًا من التقريرات assertions المحدَّدة التي يمكن التحقُّق منها بشكل فردي، والتي من خلالها يتم التحقُّق من صحة البرنامج بأكمله بسهولة"**(Turing,1949,p.67.) يعني ذلك أن التحقُّق من صحة البرامج عند تورين يتطلب تقديم تقريرات محددة وليست عامة ليتم التحقق من كلٍّ منها على حدة، ثم التحقُّق من صحة البرنامج بأكمله؛ لأن التقريرات هي جزء من البرنامج، فإذا ثبت صحة الجزء (التقريرات) كان الكل صحيحًا (البرنامج). وقد طبق تورين هذا التحقُّق على برامج المخططات الانسيابية(*) flowchart. (Wiedijk,2016,p.1.).

ب- **إسهام روبرت فلويد**:

يُعد روبرت فلويد أول من اقترح طريقة صورية formal method لإثبات صحة برامج المخططات الانسيابية التي تُعرَف باسم طريقة التقريرات الاستقرائية inductive assertions في ورقته البحثية **إسناد المعاني إلى البرامج"** assigning meanings to programs (1967) والتي ركز فيها فلويد على طرق إثبات صحة البرامج؛ من خلال التفكير في التقريرات بإعطائها الوصف الخاص بها لفهم معنى البرامج (Apt, & Olderog, 2019,p.755).إذ يقول: **"هذه الورقة محاولة لتوفير أساس كافٍ للتعريفات الصورية Formal Definitions لمعاني البرامج بلغات برمجة محددة ومناسبة، بحيث يتم وضع معيار صارم لإثبات صحة برامج الحاسوب؛ وذلك من خلال إثبات خصائص البرنامج بواسطة وضع شروط على كل أمر من البرنامج، وهذا الشرط يضمن أنه عندما يتم الوصول إلى أمر، فإن القضايا –التقريرات– المرتبطة به تكون صادقة"** .(Floyd,1967,p.19) أي أن إثبات صحة البرنامج تتطلب وضع شروط للتقريرات بوصفها خصائص للبرنامج تمنحه معنًى.

يتضح إذًا أن فلويد اهتمّ بالجانب الدلالي للغة البرمجة وأوضح أهميتها، من حيث إنها تعطي معنًى للبرنامج والتي ترتكز على صحة البناء النحوي الصحيح للعبارات، يقول فلويد: **"التعريف الدلالي للغة البرمجة في نهجنا تأسَّس على تعريف نحوي، فيجب أن تحدد أي العبارات في البرنامج الصحيح تركيبيًّا تمثِّل الأوامر، وما الشروط التي يجب أن تُفرض على تفسيرinterpretation كل أمر برمجي"**(Floyd,1967,p.20) .

(*) يُعد **النمط** في برمجة الحاسوب هو عبارة عن سلسلة الأكواد، تهدف إلى تنفيذ البرنامج ويمكن استخدامه في برنامج لأداء مهمة معينة **عدة مرات**؛ مثل: استدعاء نمط لضرب رقمين، أو إضافة رقمين، وما إلى ذلك بشكل متكرر. وتُعرف أحيانًا باسم الدالة. **انظر**(Geeksforgeeks,2021)

(*) تُعد المخططات الانسيابية من الأدوات القوية بجانب الخوارزميات في تعلُّم البرمجة للمبتدئين؛ فهي تساهم في توضيح حل خطوات المشكلة. يُوصَى دائمًا بكتابة الخوارزمية أولًا ورسم مخطط انسيابي لها، يعود التصميم الأول للمخطط الانسيابي إلى عام (1945) والذي صممه جون فون نيومان John Von Neumann. إنها أداة برمجة شائعة الاستخدام. تُستخدَم في فهم تنسيق وتسلسل العمليات التي يتم إجراؤها. ومن أهم مميزاتها: أنها وسيلة فعَّالة في منطق البرامجPrograms logic ، تتسم بسهولتها وفاعليتها في تحليل المشكلة، وأيضًا تُسهِّل عملية صيانة البرنامج أو النظام، تُستخدم كمساعدات في إظهار الطريقة التي سيعمل بها البرنامج المقترَح، وكوسيلة لفهم عمليات البرنامج الحالي. **انظر**(Walia,n.d,pp2,4; Press, 2002.p.219).

ترتكز الأسس التي اعتمد عليها فلويد في وضع شروط البرنامج، وتقديم تفسيرات للعلاقة التي تربط بين أوامر البرنامج والتقريرات على منطق الدرجة الأولى، ففي ذلك يقول: **"عند تقديم تفسير صحيح للبرنامج يجب أن نضع في اعتبارنا النسق الاستنباطيdeductive system الذي يتضمَّن البديهيات وقواعد الاستدلال لمنطق الدرجة الأولى (حساب المحمولات)"** .(Floyd,1967,p.22) ولقد طبَّق فلويد ذلك النسق على البرامج التي يتم برمجتها من خلال لغة المخططات الانسيابية (Floyd,1967,p.20).

3- التطور

طوَّر **"ريتشارد هور"** عام 1969 منطق هور بناءً على الأعمال الرائدة لعالم الحاسوب **"روبرت فلويد"**؛ حيث وضع نسقًا استدلاليًّا منطقيًّا يُسمَّى **"حساب هورHoare calculus "** الذي يُوفِّر إطارًا مناسبًا للتحقُّق من البرامج من خلال المنطق؛ حيث أشار إلى ذلك باعتباره هدفًا أساسيًّا وذلك في ورقته البحثية" **الأساس البديهي لبرمجة الحاسوب** (1969م) An Axiomatic Basis for Computer Programming، إذ يقول: **"هذا البحث محاولة لاستكشاف الأسس المنطقية لبرمجة الحاسوب باستخدام التقنيات التي تنطوي على توضيح مجموعة من البديهيات، وقواعد الاستدلال التي يمكن استخدامها في البرهنة على خصائص برامج الحاسوب"** (Hoare,1969,p.576).

إذًا سعى **"هور"** في برمجة الحاسوب إلى وضع أسس منطقية تخدم هدفه المنشود للوصول إلى آلية التحقُّق من صحة البرامج؛ للتأكُّد من أن البرنامج يحقِّق وظيفته، فإذا لم يحققها تُعتبر الأسس المنطقية للبرنامج غير صحيحة، يقول هور: **"فإن الغرض من هذه الدراسة هو توفير أساس منطقي لبراهين proofs خصائص البرنامج. ومن أهم خصائصه ما إذا كان ينفِّذ وظيفته المقصودة أم لا"** (Hoare,1969,p.576).

3-التعريف

عرف العلماء منطق هور بأنه منطق لاستنتاج reasoning صحة البرنامج، فهو يعبِّر عما ينوي البرنامج أدائه من خلال ربط قيم متغيراته قبل وبعد تنفيذ البرنامج ومتى يتوقع أن ينتهي، ويتم معرفة ما يفعله مثل هذا البرنامج بالكامل من خلال علاقة ثنائية للحالات الأولية initial (المدخلات) والنهائيةfinal (المخرجات) للبرنامج على التوالي. يتم التعبير عن مثل هذه العلاقة الثنائية " binary relation " بما يُسمَّى التقريرات المنطقية "logical assertions" (Müller, Schaefer,2018,p.119) التي تصف حالة البرنامج قبل وبعد التنفيذ؛ لذلك يُطلق عليها أيضًا **"مواصفات البرنامج"** program specification.

إن أولى الخطوات للتحقُّق من صحة البرنامج هو تكليف **Assignment** البرنامج لما يُتوقَّع القيام به، فهو عبارة عن تحديد مهمة البرنامج من خلال تحديد الشروط المسبقة**pre- conditions** (الحالات الأولية) والشروط اللاحقة **"post-conditions** (الحالات النهائية)؛ وذلك من خلال وصف لمدخلات ومخرجات البرنامج فعندما نختبر برنامجًا نختبر ما إذا كان يفي بمواصفاته أم لا. (Prasetya,2012.p.8.). حيث **"تعتمد القدرة على إثبات صحة البرامج بشكل حاسم على الصياغة الصارمة للشروط اللاحقة والشروط المسبقة، أي القدرة على إعطاء مواصفات صارمة لمهمة البرنامج"**(.Gries,1978, p.31).

يجب أن تتسم هذه المواصفات بعدم الغموض؛ وذلك من خلال استخدام لغة مفهومة بالنسبة إلى الآلة، فاللغات البشرية غير مناسبة لكتابة مواصفات البرنامج (.Prasetya,2012.p.8)، كما أن هناك العديد من أنواع اللغات بعضها أكثر ملاءمة لأنواع معينة من المشكلات" (.Crăciun,2017,p.6) وبدلًا من ذلك

يجب استخدام ما يُسمَّى **بلغة المواصفات**(*) Specification language؛ لأن لغات البرمجة تُفضِّل مثل هذه اللغة؛ لما تتضمَّنه من تركيب صارم يساعد على تجنُّب معظم المشكلات في البرمجة، ولقد اخترع هور الأسلوب الأصلي لهذه اللغة عام 1969"(Prasetya,2012.p.8.).

ثانيًا: ثلاثية هور Hoare triple

1- التعريف:

يشكِّل البرنامج والتقريرات ما يُسمَّى ب **"ثلاثية هور"** وسُمِّيت بهذا الاسم نسبة إلى مؤسسها ريتشارد هور، إلا أنه أرجع ظهور هذه الثلاثية إلى أستاذه "فلويد"، وأشار إلى أن الاختلاف يكمن في طريقة التطبيق، يقول هور: **"هذا يرجع أساسه إلى فلويد ولكن يتم تطبيقه على النصوص بدلًا من المخططات الانسيابيةflowcharts"** (Hoare,1969,p.577). إذًا طبَّق هور هذه الثلاثية على البرمجة النصية، في حين طبَّق فلويد الثلاثية على المخططات الانسيابية.

2- الصياغة الرمزية:

يُعبِّر عن ثلاثية هور من خلال الصياغة التالية: (Hoare,1969,p.577).

$$\{p\}\ C\ \{Q\}$$

الشرح والتوضيح: -

P: تشير إلى الشروط المسبقة.

Q: تشير إلى الشروط اللاحقة.

C: تشير إلى البرنامج (عبارات البرنامج)Program statement

القراءة: تكون الصياغة التالية { P } C { Q } صحيحة **جزئيًّا Partial Correctness**إذا كانت **تستلزم** التالي:(Hoare,1969,p.577).

- إذا كانت الشروط المسبقة **"P"** صادقة قبل تنفيذ البرنامج **"C"**، وإذا **انتهى** تنفيذ **"C"**.

- إذًا ستكون الشروط اللاحقة **Q** صادقة في الحالة التي يتم فيها تنفيذ **C.**

بمعنى آخر: "إذا كانت متغيرات البرنامج تستوفي الشرط P وتم تنفيذ البرنامج C إذًا فإن متغيرات البرنامج تستوفي الشرط Q" .(Poroor, 2021,p.1)**.**

"هذا ما يُعرَف **"بالصحة الجزئية"** { P } C { Q } فليس من الضروري إنهاء تنفيذ C عند البدء في استيفاء حالة الشرط المسبق P ويتطلب ذلك في حالة أن يكون Q معلقًا على إنهاء تنفيذ **البرنامج.** في حين تتطلب **الصحة الكلية**A total correctness إنهاء تنفيذ البرنامج C بالكامل"، ويُعبَّر عنها

(*) تُعد لغة المواصفات جزءًا لا يتجزأ من تطوير البرامج، تُحدِّد المواصفات دلالات البرنامج من خلال لغة ذات أسس منطقية معروفة ومحددة. وهي لغة رسمية في علوم الحاسوب تُستخدم في تحليل وتصميم الأنظمة لوصف نظام بمستوى أعلى بكثير من لغة البرمجة. **انظر**(Tarkoma , 2003, p.1).

من خلال الثلاثية [Q] C [P]" (Gordon,1988,p.5) فتشير هذه الأقواس إلى الصحة الكلية [ ]، في حين الصحة الجزئية يُشار إليها بالأقواس { }.

بمعنى آخر الصحة الكلية تعني إذا كان البرنامج يستوفي حالة الشروط المسبقة؛ إذًا سينتهي تنفيذ البرنامج، علاوة على ذلك ستكون الشروط اللاحقة صادقة وستصل إلى حالتها النهائية. في حين الصحة الجزئية تعني إذا كان البرنامج يستوفي حالة الشروط المسبقة؛ إذًا سيتم تنفيذ البرنامج وعندئذ لا يتوقف تنفيذ البرنامج، وإذا انتهى تنفيذه فإن الشروط اللاحقة ستصل إلى حالتها النهائية (Almeida,2011,p139).

تُستخدَم ثلاثية الصحة الجزئية لتحديد سلوك البرامج والاستدلال عليه؛ حيث C هو أمر البرنامج، أما Q وP هما محمولات الحالة، حيث P يصف الحالة الأولية للبرنامج وQ يصف الحالة النهائية (Svendsen,2016,p.8. ; Nielson,1984,p.10)، فإذا كان تنفيذ البرنامج C بدأ في أي حالة للذاكرة memory state تفي بالشرط المسبق، وانتهى هذا التنفيذ إذًا سيكون الشرط اللاحق صحيحًا عند الانتهاء. (Cousot,1990.p2.).

إذًا تستلزم صحة الصياغة صدق الشرط المسبق أولًا وتنفيذ البرنامج الذي يلزم عنهما صدق الشروط اللاحقة، ويمكن أن نُمثِّل هذه العلاقة من خلال هذه الصياغة الرمزية التي تُعبِّر عن الصحة الجزئية للثلاثية (Barthe,2020.p.15):

$$|= \{P \wedge C\} \supset Q$$

مما سبق يمكن القول بأن ثلاثية هور تقوم على فكرة **الاستلزام المفهومي** Intentional Implication لدى إيفريت نيلسون (Nelson, E. J (1900 – 1988) (عبد الجواد، أحمد 2022، ص ص163-168)؛ تعتمد صحة هذه الثلاثية على المعنى الذي يربط بين أجزاء الثلاثية وليس على قِيَم صدقها، فإذا كانت هذه الثلاثية تقوم على اللزوم المادي أو الدقيق سيؤدي ذلك إلى خطأ في الثلاثية مما يؤدي إلى مفارقة؛ لأن من ضمن حالات صدق اللزوم المادي كذب المقدم وصدق التالي، الذي ينتج عنه عدم صحة هذه الثلاثية، فصحة الثلاثية تبدأ في الأساس من صدق الشروط المسبقة، فلا يمكن أن تكون الشروط المسبقة كاذبة والشروط اللاحقة صادقة. إذًا ثلاثية هور تقوم على العلاقة القائمة بين التقريرات المنطقية P, Q، وليس على قيم صدقها.

3- التقريرات المنطقية:

أ- التعريف:

عرَّف العلماء التقريرات بأنها تراكيب لغوية linguistic constructions تسمح بالتعبير عن خصائص البرنامج، وتُستخدَم في سياقات مختلفة ولأغراض مختلفة تتعلق بتطوير البرنامج (Puebla & Bueno,2000,p.23). فهي ضمن خصائص متغيرات البرنامجProgram Variables . واعتبرها علماء آخرون أنها صيغ منطقية من الدرجة الأولى تُستخدَم في ربط القِيم قبل وبعد تنفيذ البرنامج (Müller, Schaefer,2018,p.119) وذلك من أجل تيسير عملية فهم البرنامج والتحقق منه؛ حيث يهدف التحقق من البرنامج وتصحيحه القائم على التقريرات إلى التفكير تلقائيًا في صحة البرنامج عن طريق التحقُّق مما إذا كانت التقريرات صحيحة أم لا لبرنامج معين؛ وذلك بواسطة نسق استنباطي ; (Deransart et al., 2000,p.6. Drabentt & Matuszyfiski, 1987,p.169)

ب- مكوِّنات التقريرات:

تتكوَّن التقريرات من: شرط مسبق، وشرط لاحق وكلاهما يعبِّر عن محمولات تصف البرنامج وتحدد خصائصه (Hoare,1969,p.577). الشرط المسبق محمول يصف الشرط الذي تعتمد عليه الدالة للتشغيل الصحيح للبرنامج، ويجب على العميل استيفاء هذا الشرط، في حين الشرط اللاحق عبارة عن محمول يصف الحالة التي تُنشئها الدالة بعد التشغيل الصحيح، ويمكن للعميل الاعتماد على أن هذا الشرط صحيحًا بعد استدعاء الدالة (Aldrich, 2013,p.1). إذًا "تُستخدَم التقريرات أيضًا لوصف السلوك الفعلي للبرنامج لهذا تُعرَف بمواصفات البرنامج.

ج-أنواع التقريرات:

على غرار وجود نوعين من القضايا: قضايا ذرية وقضايا مركبة. "هناك نوعان من التقريرات في منطق هور: تقريرات ذريةatomic assertions وتقريرات مركَّبة compound assertions التقريرات الذرية العبارات التي تتكوَّن من **الحدود** -التي تشير إلى **قيم ثابتة** مثل الأرقام (1، 4+5) وقد تشير إلى متغيرات مثل(X,Y,Z) التي قد تتغيَّر قيمتها- **والمحمولات** التي يمكن الحكم عليها بالصدق أو الكذب ⊥ T, (Pharabod , 2017,p.16) أي أن التقريرات الذرية هي العبارات الذرية التي يتم تكوينها باستخدام الحدود والمحمولات معًا.

أمثلة:

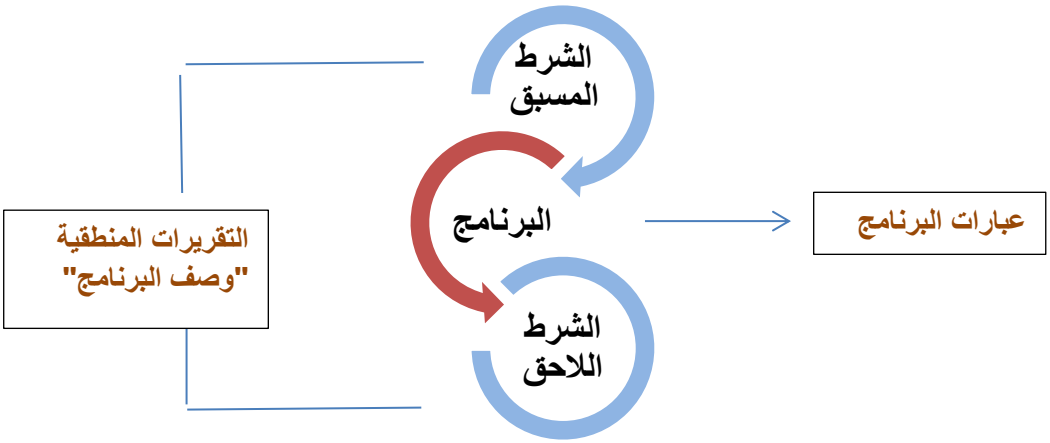
- X = 1
- PRIME (3)
- ODD(X)

الشرح: الأمثلة السابقة هي تقريرات ذرية يمكن الحكم عليها بالصدق أو الكذب، تتألَّف من حدود ومحمولات. تُعتبر **PRIME** و**ODD** هي أمثلة على المحمولات. والمتغير "**X**"والقيمة الثابتة "**3**" هما أمثلة على الحدود(Gordon, 2012,p.14.).

التقريرات المركبة يتم إنشاؤها من التقريرات الذرية باستخدام: الروابط المنطقية (العطف، الفصل، اللزوم، النفي) أو استخدام التسوير مثل السور الكلي (∀ س) أو السور الوجودي (E س) , Pharabod) 2017,p.16).

مما سبق يتضح أن الشروط المسبقة والشروط اللاحقة تُسمَّى "بمواصفات البرنامج" أو "التقريرات المنطقية" والتي مهمتها وصف البرنامج من خلال المدخلات والمخرجات، **ويمكن القول إن المواصفات هي قضية تصف حالة البرنامج قبل وبعد تنفيذ البرنامج**، أما البرنامج نفسه فيحتوي علي عبارات تشير إلى تنفيذه والهدف منه.

شكل (1) "ثلاثية هور"



4- كتابة المواصفات

تتطلَّب كتابة المواصفات استخدام الرموز والروابط المنطقية، وأشار عالم الحاسوب البريطاني "مايك جوردون Mike Gordon (1948- 2017 م) في كتابه الأكاديمي "القراءة الخلفية في منطق هور Background reading on Hoare Logic" أن: **"كتابة شروط متغيرات البرنامج تتم باستخدام الروابط المنطقية:** ¬ النفي، ∧ العطف، ∨ الفصل، → اللزوم، ↔ التكافؤ**"** Gordon, (2012,p.10.).

يقول هور: **"يمكن تحديد الوظيفة المقصودة من البرنامج أو جزء من البرنامج عن طريق إجراء تقريرات عامة general assertions حول القيم التي ستأخذها المتغيرات بعد تنفيذ البرنامج، ولن تنسب هذه التقريرات عادةً قيمًا معينة لكل متغير، ولكنها ستحدد خصائص عامة للقيم والعلاقات التي تربط بينها، ونستخدم رموز المنطق الرياضي للتعبير عن هذه التقريرات"** (Hoare,1969,p.577).

إذًا متغيرات البرنامج لا تأخذ قيمًا محددة بعينها، ولكن يتم وضع خصائص لهذه القيم من خلال الشروط المنطقية باستخدام رموز المنطق الرياضي.

5- الفرق بين متغيرات البرنامج والمتغيرات المنطقية:

يُحيلنا الحديث عن المتغيرات نحو توضيح الفرق بين متغيرات البرنامج program variables والمتغيرات المنطقية. متغيرات البرنامج هي المتغيرات المستخدَمة في البرنامج، وتشير إلى مواقع الذاكرة memory locations في المتجر (مخزن الحاسوب) وستتغير قيمتها في أثناء تنفيذ البرنامج، أي أنها تحدث في أوامر فعلية للبرنامج C ويمكن أن تحدث أيضًا في التعليقات التوضيحية في الشروط المسبقة والشروط اللاحقة، ولكن دون استخدام الأسوار الكمية لذلك هي متغيرات حرة. أما **المتغيرات المنطقية** فتشير إلى قيم ثابتة لا تتغير في أثناء تنفيذ البرنامج، تُستخدَم لتذكُّر remember **قيم** متغيرات البرنامج في مرحلة معينة من التنفيذ للرجوع إليها لاحقًا، وهي متغيرات مقيَّدة bound variables لأنها تُحدِّد بعض الشروط المسبقة واللاحقة بالأسوار الكمية، أو في التعليقات التوضيحية الواردة في البرنامج (Capretta, 2010,p.2 ; Almeida & Frade, 2011,p.139).

إذًا المتغيرات المنطقية في منطق هور تُستخدَم كثوابت، فالمتغيرات تختلف عن الثوابت على أساس **"صفة بنائية"** وهي صفة القابلية للإبدال، فالمتغيرات في أي نسق تكون خاضعة للتبديل طبقًا لقواعد هذا

النسق، في حين الثوابت لا يمكن تبديلها؛ أي الفارق بين المتغير والثابت ليس في كون الثابت له معنًى محدد بينما لا يكون للمتغير معنى، بل إن الفارق بينهما أن الثابت لا يمكن إبداله، بينما المتغير هو الذي يكون خاضعًا للتبديل(*) (ا.د/ سهام النويهي، د.ت، ص81).

6- مثال تطبيقي:

كود (1)

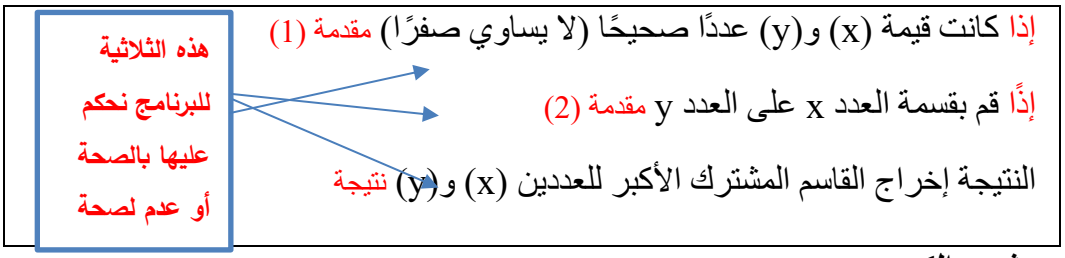
"كود برمجي مكتوب بلغة المواصفات بنمط ثلاثية هور"

```
{* x > 0 ∧ y > 0 *}
Euclid (x, y)
{* return = gcd x y *}
```

المصدر: Prasetya,2012, p 9.

القراءة:

يُقرأ الكود السابق كالتالي (Prasetya,2012.p.9.)



شرح الكود:

تُسمَّى التعبيرات الواقعة بين {* *} بالتقريرات **المنطقية** التي تُعتبر من مواصفات البرنامج التي تتمثَّل في الشرط المسبق والشرط اللاحق(Prasetya,2012.p.8.).

(*) تتضح تلك السمة للمتغيرات والثوابت أيضًا في لغات البرمجة، حيث استخدام الحروف الهجائية لتمثيل القضايا ليس متغيرات بالمعنى الحقيقي، فهي ببساطة تُمثِّل القضايا والعبارات بشكل رمزي وتختلف عن متغيرات منطق المحمولات ومتغيرات لغات البرمجة عالية المستوى مثل لغة "C" و"Fortran"؛ حيث يشير المتغير في لغة البرمجة إلى مجال من **القِيَم**، على سبيل المثال المتغير قد يكون عددًا صحيحًا في برنامج "Fortran" ؛ حيث يرمز المتغير إلى أي رقم صحيح وفقًا لمواصفات كل لغة، أي أن المتغير يعبّر عن قيمة تُعتبر من الثوابت وهي الواحد الصحيح. **انظر**:(Chowdhary.2020, p.28).

مكوّنات ثلاثية هور:

يحتوي الكود السابق على عناصر ثلاثية هور، كالتالي[8]:

- {* x > 0 ∧ y > 0 *} يمثِّل الشرط المسبق.
- Euclid (x,y) يمثِّل عبارات البرنامج.
- {* return = gcd x y *} يمثِّل الشرط اللاحق.

خطوات البرنامج: هناك مجموعة من الخطوات المنطقية في تنفيذ البرنامج طبقًا لثلاثية هور:

1- يُدخِل المستخدم قيمةx وy.
2- يتم فحص هذه القِيَم إذا كانت تستوفي الشروط المسبقة المخزَّنة في ذاكرة الحاسوب أم لا.
3- إذا كانت القيم التي أدخلها المستخدم تستوفي الشروط المسبقة؛ إذًا يتم تنفيذ البرنامج - أي قسمة قيمة العدد x على قيمة العدد y.
4- بمجرد تنفيذ البرنامج، تتحقَّق الشروط اللاحقة التي تتمثل في إخراج القاسم المشترك الأكبر لعددين صحيحين.

نوع التقريرات: التقريرات في المدخلات (الشروط المسبقة) تقريرات مركَّبة تتألف من تقريرات ذرية، ورابطة العطف التي تربط بين القضية الأولى x> 0 والقضية الثانية y > 0.

متغيرات البرنامج: هي قيمة كلٍّ من x,y.

الدالة: Euclid ونوعها دالة ثنائية **تتمثل وظيفتها** في قسمة قيمة x على قيمة y.

المحمولات: هي مواصفات البرنامج والتي تتمثَّل في الشروط المسبقة والشروط اللاحقة.

- المحمول في المدخلات " x > 0 ∧ y > 0" هو وصف المتغيرات بأنها أكبر من الصفر، أي أن كلاهما عدد صحيح لا يساوي الصفر.
- المحمول في المخرجات هو Gcd حيث يصف ويُحدِّد حالة المخرجات في الشروط اللاحقة، المتمثل في إخراج القاسم المشترك الأكبر لعددين صحيحين من yوx. ومن خلال هذه المحمولات استطعنا أن نحدد وظيفة وهدف البرنامج.

يمكن القول إذًا إن ثلاثية هور تشير إلى **"قيمة منطقية"** فإذا كانت القيمة صادقة نقول إنها ثلاثية صحيحة، وإذا كانت خاطئة نقول إن الثلاثية غير صحيحة (Backhouse, 2002,P.p.6 -7)؛ لذلك يُعتبر منطق هور ثنائي القيم؛ لأننا نحكم على الشروط المسبقة والشروط اللاحقة من خلال الصدق أو الكذب وبناءً عليه نحدد صحة الثلاثية أو عدم صحتها.

جدول (1)
الفرق بين الثلاثية الصحيحة وغير الصحيحة

ثلاثية صحيحة	ثلاثية غير صحيحة
{i = 0} i:= i+1 { i = 1}	{i =1} i:= i+1 { i = 0}
هذه الثلاثية صحيحة: حيث قيمة الشروط اللاحقة صادقة، وتلزم بشكل صحيح عن تنفيذ البرنامج، والشروط المسبقة التي هي أيضًا صادقة.	هذه الثلاثية غير صحيحة؛ حيث قيمة الشروط اللاحقة كاذبة ولا تلزم بشكل صحيح عن تنفيذ البرنامج، والشروط المسبقة التي قد تكون كاذبة.

(Backhouse, 2002,P.p.6 -7)

[8] - حاولت الباحثة شرح الكود استنادًا إلى فهمها لأسس منطق هور.

ثالثًا: قواعد النسق الاستنباطي

يُعتبر هور أول من أسس نسقًا استنباطيًّا **deductive system** لهذا المنطق، لكنه لم يُبدع فيه من فراغ بل حاول تطوير الأفكار التي طرحها فلويد (Hoare,1971,pp. 334-341). من أساسيات النسق الاستنباطي والتي تمثلت لدي هور في إثبات صحة الثلاثية بواسطة البديهيات، وقواعد الاستدلال الصحيحة، وفي ذلك يقول:

> **" تُعد برمجة الحاسوب علمًا دقيقًا حيث اكتشاف جميع خصائص البرنامج وجميع عواقب تنفيذه.. يتم من خلال نص البرنامج نفسه text of the program عن طريق التفكير الاستنباطي البحت purely deductive reasoning. يتضمن التفكير الاستنباطي تطبيق قواعد الاستدلال الصحيحة على فئة من البديهيات الصحيحة valid axioms لذلك من المرغوب والمثير للاهتمام توضيح البديهيات وقواعد الاستدلال التي تكمن وراء استنتاجنا لبرامج الحاسوب**"(Hoare,1969,p.576).

ويشترط أن تكون التلاتية قابلة للبرهان **{P} C {Q}** ⊢ **{P} C {Q}** لكي نتبت صحتها **|= { P} C {Q}** (Attie,2001,p.36)البرهان هو سلسلةsequence من الأسطر، كلٌّ منها إما بديهية منطقية أو يتبع من الأسطر السابقة من خلال قاعدة الاستدلال المنطقي (Gordon, 2012,p.18.).

يكمن السبب في بناء البراهين في محاولة التأكد من استخدام طرق الاستدلال الصحيحة التي تتمثَّل في صحة كلٍّ من البديهيات وقواعد الاستدلال، حينها يمكن أن يكون المرء واثقًا من صحة الاستدلالات، هذا من جانب وعلى الجانب الأخر إذا كانت أيٌّ من البديهيات أو قواعد الاستدلال غير سليمة سيؤدي ذلك إلى نتائج خاطئة (Gordon, 2012,p.18.).

1- البديهيات:

تُمثِّل البديهيات لدى هور الأساس في نسقه الاستنباطي للبرهنة على صحة البرامج، فالنهج البديهي لديه لا غنى عنه لتحقيق موثوقية البرنامج، وقابلية التطبيق العملي وقابلية النقل ولإثبات وتوثيق البرنامج وتعديله، بالإضافة إلى أنه يمكن صياغة البديهيات بطريقة مستقلة إلى حد كبير عن بعضها البعض، بحيث يمكن العمل بحرية على بديهية واحدة أو مجموعة من البديهيات دون خوف، فمن السهل نسبيًّا تكوين البراهين من البديهيات، وهذا سيكون أفضل من لغة بها العديد من البديهيات الغامضة التي يصعب فهمها (Hoare,1969, pp.583-597). وقد قدم هور نوعان من البديهيات؛ هما: بديهية التخطي **skip axiom**، وبديهية التكليف **Assignment Axiom.**

أ- بديهية التخطي:

- **التعريف**

هذه البديهية ليس لها تأثير في البرامج؛ حيث القيمة المخزَّنة في الذاكرة بعد التنفيذ هي نفسها التي كانت قبل التنفيذ (Capretta, 2010,p.1)؛ إذًا هذه البديهية لا تُغيِّر حالة البرنامج فكل ما ينطبق قبل التخطي يظل أيضًا صحيحًا بعد ذلك.

- **الصياغة الرمزية:**

يُعبِّر عن بديهية التخطي من خلال الصياغة (D'Souza, 2012,p.10):

{P} skip {P}

*** مثال:**

{5} skip {5}

يلاحظ أن قيمة المدخلات هي ذاتها قيمة المخرجات {5} لأن البرنامج قائم على بديهية التخطي التي لا تؤثر في مخرجات البرنامج.

ب- بديهية التكليف

التعريف: يُقصد بالتكليف في البرمجة عبارةstatement يتم استخدامها لتعيين أو تخصيص قيمة إلى اسم متغير. يتم الإشارة إلى العامل المُستخدَم للقيام بالمهمة بعلامة يساوي (=)... من الممكن أن يحتفظ المتغير نفسه بقيم مختلفة في أوقات مختلفة" (Rouse. 2016)

إذًا بديهية التكليف ليس لها فرضية. بمعنى آخر، يمكن اعتبار أي مثيل لبديهية التكليف صالح دون الحاجة إلى إثبات صحة الفرضية أولًا" (Attie,2001,p.39)؛ لذلك فهي تُعبِّر عن العبارات البرمجية التي تتساوى فيها القيم بين الجانب الأيمن والجانب الأيسر لنُسلِّمَ بصحتها دون الحاجة إلى برهان أو دليل. والجدير بالذكر أن المعامل المُستخدَم في برمجة الحاسوب هو **رمز الهوية** (=) في منطق الدرجة الأولى؛ حيث يشير هذا الرمز إلى تساوي الأشياء في القيمة نفسها.

ضع في اعتبارك عبارة التكليف التالية (Hoare,1969, p577):

x: =f

X: ترمز إلى متغير بسيط.

F: هي تعبير عن لغة برمجة تحتوي على x.

يعني هذا أن المتغير x له القيمة نفسها للتعبير f فهما يحملان القيمة ذاتها؛ من خلال رمز الهوية الذي يشير إلى تساوي القيم. حيث قيمة X في الحالة النهائية (المخرجات) هي قيمة F في الحالة الأولية (المدخلات).

الصياغة الرمزية:

يُعبِّر عن بديهية التكليف بواسطة الصياغة الرمزية (*)(Gordon, 2012,p.20) **:**

$$\vdash \{P\ [E/V]\}\ V:=E\ \{P\}$$

القراءة: تشير V إلى أي متغير، وE أي تعبير منطقي، وP أي عبارة، ويشير الرمز [E/V] P إلى استبدال التعبير E بجميع تكرارات المتغير V. (Gordon, 2012,p.20).

(*)هناك شكل آخر للصياغة تُعبِّر عن بديهية التكليف لمنطق هور $\{Q(e)\}\ x := e\ \{Q(x)\}$.

القراءة: افترض أن Q(x) هي عبارة عن محمول يتضمن المتغير X وأن Q (e) تشير إلى الصيغة نفسها مع استبدال التعبير e بجميع تكرارات المتغير x **انظر:** ((Gore,2016.p.27.

تُمثِّل بديهية التكليف حقيقة أن قيمة المتغير V بعد تنفيذ أمر التكليف V:=E تساوي قيمة التعبير E قبل تنفيذها(Gordon, 2012,p19) .

الشرح: البديهية السابقة تُسمَّى بديهية التكليف؛ لأنها تحتوي على أمر التخصيص V: =E. فالمتغير V تم تخصيصه بقيمة تساوي E، من ثمَّ من الممكن استبدال قيمة التعبير E بكل متغير موجود.

أكد هور على أن الصياغة الرمزية لبديهية التكليف ليست بديهية محددة بعينها، بل هي مخطط بديهي axiom schema(*) يصف مجموعة لا نهائية من البديهيات التي تشترك في نمط واحد، ويوصَف هذا النمط بحدود تركيبية بحتة purely syntactic terms؛ لذلك إذا كان نص البرنامج يتوافق مع هذا النمط فهو مؤهل كبديهية، والتي تظهر بشكل صحيح في أي سطر من البرهان (Hoare,1969, p577).

مثال تطبيقي:

إن أفضل طريقة لحل مسائل البديهيات لاشتقاق الشروط المسبقة والتأكد أن هذه البديهية صالحة. "ليس بالبدء من الشرط المسبق والاتجاه نحو الشرط اللاحق (Gore,2016,p.29). إنما التوجه نحو ما وضعه عالم الحاسوب الهولندي أيدسكر ديكسترا Edsger W. Dijkstra(1930 -2002) لما يُعرف بإستراتيجية ديكسترا للخلف"(Bernot,2019,p.146) Dijkstra Backward strategy. و التي مفادُها أن تبدأ بهدفك من (الشرط اللاحق) وانتقل إلى الخلف backwards ثم خذ الشرط اللاحق وانسخه إلى الشرط المسبق، ثم استبدل بكل تكرارات المتغير **V** التعبير **E** (Gore,2016,p.29).

مثال: ضع في اعتبارك الكود البرمجي X:= 2 وافترض أن الشرط اللاحق المطلوب هو (Y=X) يمكن العثور على الشرط المسبق من خلال الخطوات التالية (Gore,2016,p.30)**:**

1- نقوم بكتابة المعطيات على شكل ثلاثية هور.

{ ..... } X:= 2 {Y = X}

2- نبدأ من الشرط اللاحق ثم ننتقل إلى الخلف.

{ ..... } X:= 2 {Y = X}

3- نقوم بنسخ الشرط اللاحق إلى الشرط المسبق.

{Y = X} X:= 2 {Y = X}

4 – نستبدل التعبير 2 بكل تكرارات المتغير X.

{Y = 2} X:= 2 {Y = X}

5 – النتيجة النهائية لشكل الثلاثية الصحيحة |= {Y = 2} X:= 2 {Y = X}

(*) **المخطط** هو "نموذج" أو "إطار" أو "نمط" يُستخدَم لتحديد عدد لا نهائي محتمَل من العبارات أو الجمل، وتُستخدم المخططات في المنطق لتحديد قواعد الاستدلال، وفي الرياضيات لوصف النظريات التي تحتوي على عدد لا نهائي من البديهيات وفي الدلالات لإعطاء شروط ملائمة لتعريف الصدق. **انظر**:(Corcoran & Hamid,2022).

2- قواعد الاستدلال

يتطلب العلم الاستنتاجي بجانب البديهيات وجود قاعدة استدلال واحدة على الأقل، التي تسمح باستنتاج نظريات جديدة من واحدة أو أكثر من البديهيات أو النظريات، التي تم إثباتها بالفعل وتأخذ قواعد الاستدلال هذا الشكل (Hoare,1969, p578):

$$\text{If } \vdash X \text{ and } \vdash Y, \text{ then } \vdash Z$$

ويُعبّر عن قواعد الاستدلال من خلال هذا الشكل:

$$\frac{\vdash\{P_1\}\ Q_1\{R_1\}\ \ldots.\vdash\{P_n\}Q_n\ \{R_n\}}{\vdash\{P\}Q\ \{R\}}$$

القراءة: إذا كانت ثلاثيات هور $\vdash\{P_1\}\ Q_1\{R_1\}\ \ldots.\vdash\{P_n\}Q_n\{R_n\}$ يمكن إثباتها provable في النسق البرهاني، فإن $\vdash\{P\}Q\ \{R\}$ يمكن أيضًا إثباتها.

تنص قواعد الاستدلال على أنه إذا تم إثبات صحة الفرضيات (الجزء الموجود أعلى السطر)، فعندئذ الاستنتاج (الجزء الموجود أسفل السطر) صالح أيضًا.(Attie,2001,p.36).

1- قاعدة النتيجة Consequence Rule

التعريف والهدف:

في بعض الأحيان، لن تكون الشروط المسبقة والشروط اللاحقة التي نحصل عليها، هي تلك التي نريدها تمامًا في الموقف المعين المطروح -فقد تكون متكافئة منطقيًّا ولكنها مختلفة من الناحية التركيبية– فقد لا تتماشي مع الهدف الذي يتم إثباته، أو قد تكون في الواقع أضعف منطقيًّا (للشروط المسبقة) أو أقوى (للشروط اللاحقة) مما نحتاج إليه؛ لذلك فإن **"قاعدة النتيجة هي القاعدة التي تعمل على تعزيز الشرط المسبق وإضعاف الشرط اللاحق"** (Svendsen,2016,p.30)، فينبغي أن يكون الشرط المسبق قويًّا من الوجهة المنطقية لأنه يُعتبر أساسًا لقوة البرنامج، أما الشروط اللاحقة فلا يشترط ذلك؛ لأن صحة نتائجها تتوقف على قوة الشروط المسبقة.

الصياغة الرمزية:

تنقسم الصياغة الرمزية لهذه القاعدة إلى نوعين (Hoare,1969, p578; Prasetya,2012 ,p.197):

أ- قاعدة تقوية الشروط المسبقة Pre-condition strengthening

يُعبَّر عن هذه القاعدة من خلال الصياغة:

$$\frac{\{P\}\ Q\ \{R\}\ \wedge\ P_1 \Rightarrow P}{\{P_1\}\ Q\ \{R\}}$$

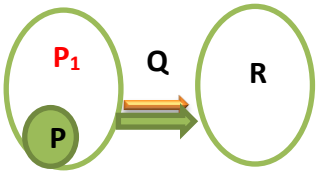
القراءة: إذا كانت P هي الشرط المسبق للبرنامج Q للوصول للنتيجة R إذًا فإن أي تقرير آخر P_1 يلزم عنه منطقيًّا P (Hoare,1969, p578). معنى ذلك أن الشرط الأقوى هو الشرط الذي يلزم عنه شيء آخر،

فإذا كانت $\mathbf{P \Rightarrow P_1}$ يعني أن P_1 أقوى منطقيًّا من P وتركيبها النحوي يحقق الهدف من البرنامج، وفيما يلي يوضح الشكل رقم (2) كيفية عمل هذه القاعدة

شكل (2)

"توضيح قاعدة تقوية الشروط المسبقة"

$\mathbf{\{P_1\}\ Q\ \{R\}}$



{P} Q {R}

المصدر: (Prasetya,2012.p.15.)

مثال: الكود البرمجي الآتي مكتوب بنمط ثلاثية هور، والمطلوب معرفة النتيجة الصحيحة لهذا الكود باستخدام قاعدة النتيجة (Gore,2016,p.42).

$$\frac{\{y > 0\}\ x := y\ \{x > 0\} \Lambda\ (y = 2) \Rightarrow (y > 0)}{\text{؟؟؟؟؟؟؟؟؟}}$$

المعطيات: مجموعة من الافتراضات القابلة للبرهان وهي:

- كود برمجي على نمط ثلاثية هور يحتوي على: (1)

شرط مسبق: $\{y > 0\}$

أمر البرنامج: $x := y$

شرط لاحق: $\{x > 0\}$

- علاقة اللزوم $(y = 2) \Rightarrow (y > 0)$؛ حيث$(y = 2)$ يلزم عنها $(y > 0)$ (2)

من (1) لـ (2) يتضح أن هذا الكود مكتوب **بقاعدة النتيجة** لمنطق هور، ومن خلال التعويض بالقاعدة التالية:

$$\frac{\mathbf{\{P\}\ Q\ \{R\}\ \Lambda\ P_1 \Rightarrow P}}{\mathbf{\{P_1\}\ Q\ \{R\}}}$$

ستصبح النتيجة الصحيحة:

$$\frac{\mathbf{\vdash \{y > 0\}\ x := y \vdash \{x > 0\} \Lambda\ (y = 2) \Rightarrow (y > 0)}}{\mathbf{x := y \vdash \{y = 2\}\ \{x > 0\}}}$$

فالشرط المسبق $\{y = 2\}$ على الرغم من أنه متكافئ منطقيًّا مع $\{y > 0\}$ فإن الشرط الأول أقوى منطقيًّا من الثاني فهو **أكثر تخصيصًا**، فنحن نختار الأقوى لتعزيز الشرط المسبق لذلك الكود:

$\{x > 0\}$ x:= y $\{y = 2\}$ أقوى منطقيًّا من $\{x > 0\}$ x:= y $\{y > 0\}$

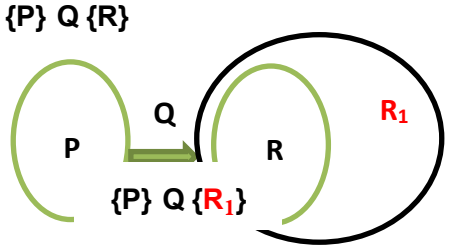
ب- **قاعدة إضعاف الشرط اللاحق.Post-condition weakening**

يُعبِّر عن هذه القاعدة من خلال الصياغة:

$$\frac{\{P\}\ Q\ \{R\} \wedge R \Rightarrow R_1}{\{P\}\ Q\ \{R_1\}}$$

القراءة: إذا كان تنفيذ البرنامج Q يستلزم صدق الشرط اللاحق R إذًا فهذا يستلزم أيضًا صدق كل التقريرات التي تلزم عن R (Hoare,1969, p578)، نستبدل الشرط اللاحق القوي بشرط لاحق أضعف، لأن الهدف هو قوة الشرط المسبق. إذا كان الشرط اللاحق الأساسي**R** يلزم عنه الشرط اللاحق $\mathbf{R_1}$ إذًا فإن الشرط الأقوى هو **R** والشرط الأضعف هو $\mathbf{R_1}$.

الشكل رقم (3)
يوضِّح قاعدة إضعاف الشرط اللاحق



المصدر: (Prasetya,2012.p.12.)

مثال: الكود البرمجي الآتي مكتوب بنمط ثلاثية هور، والمطلوب معرفة النتيجة الصحيحة لهذا الكود باستخدام قاعدة النتيجة (Gore,2016,p.42).

$$\frac{\{x > 2\}\ x:= x+1\ \{x > 3\} \wedge (x > 3) \Rightarrow (x > 1)}{??????????}$$

المعطيات: لدينا مجموعة من الافتراضات القابلة للبرهان، وهي:

- كود برمجي على نمط ثلاثية هور يحتوي على: (1)

شرط مسبق: $\{x > 2\}$

أمر البرنامج: **x:= x+1**

شرط لاحق: **{x > 3}**

- علاقة اللزوم: $(x > 1) \Rightarrow (x > 3)$ حيث $(x > 3)$ يلزم عنها $(x > 1)$ (2)

من (1) لـ (2) يتضح أن هذا الكود مكتوب **بقاعدة النتيجة** لمنطق هور ومن خلال التعويض بالقاعدة التالية:

$$\frac{\{P\}\ Q\ \{R\}\ \Lambda\ R \Rightarrow R_1}{\{P\}\ Q\ \{R_1\}}$$

ستصبح النتيجة الصحيحة:

$$\frac{\vdash \{x > 2\}\ x:= x+1 \vdash \{x > 3\}\ \Lambda\ (x > 3) \Rightarrow (x > 1)}{\vdash \{x > 2\}\ x:= x+1\ \{x > 1\}}$$

إذًا على الرغم من أن **الشرط اللاحق** $\{x > 3\}$ متكافئ منطقيًّا مع $\{x > 1\}$، فإن الشرط الثاني أضعف من الأول منطقيًّا، فهو أقل تخصيصًا، وبالرغم من صحة الشرطين فإنه يتم اختيار الشرط الأضعف ليناسب الشرط اللاحق.

2- قاعدة التكوين: Composition Rule

التعريف: "يتكوَّن البرنامج في قاعدة التكوين بشكل عام من سلسلة من العبارات التي يتم تنفيذها واحدة تلو الأخرى" (Hoare,1969, p578)؛ حيث تنطبق قواعد التكوين على البرامج التي يتم تنفيذها بالتسلسل. "يمكن فصل العبارات بفاصلة منقوطة؛ مثل: $(Q_1; Q_2;... ; Q_n)$" (Hoare,1969, p578) حيث يتم تنفيذ Q_1 قبل Q_2 وهكذا حتى يتم تنفيذ البرنامج بشكل متسلسل للوصول إلى النتيجة؛ لذلك يُطلَق عليها أيضًا قاعدة التسلسل The sequencing rule. (Frohardt, 2009,p.3).

الصياغة الرمزية: يُعبّر عن هذه القاعدة من خلال الصياغة الرمزية التالية(Hoare,1969, p578):

$$\frac{\vdash \{P\}\ Q_1\ \{R_1\}\ \Lambda \vdash \{R_1\}\ Q_2\ \{R\}}{\vdash \{P\}\ Q_1; Q_2\ \{R\}}$$

القراءة: تنص القاعدة أنه إذا كان الشرط اللاحق الذي تم إثباته في الجزء الأول من البرنامج $\{P\}$ Q_1 $\{R_1\}$ متطابقًا مع الشرط المسبق للجزء الثاني من البرنامج $\{R_1\}$ Q_2 $\{R\}$ الذي بموجبه نصل إلى النتيجة المرجوة، فإن البرنامج بأكمله (الجزء الأول والجزء الثاني) سيحقق النتيجة النهائية Hoare,1969, p578). يعني هذا أنه عندما يكون التكوين المتسلسل للعبارات البرمجية Q_1, Q_2 **فيجب أن يكون هناك شرط لاحق لـ** Q_1 **يساوي الشرط المسبق لـ** Q_2 (Swinker, 2012,p.3).

إذًا تحكم قاعدة التكوين تنفيذ البرنامج بنظام وتسلسل من خلال مجموعة من الخطوات حتى نصل إلى النتيجة الصحيحة، على النحو التالي:

1- وجود شرط مسبق صحيح للجزء الأول من البرنامج. $\{P\}\ Q_1\ \{R_1\}$

2- تتطابق نتيجة (الشرط اللاحق) الجزء الأول من البرنامج مع الشرط المسبق للجزء الثاني من البرنامج للوصول إلى نتيجة الجزء الثاني. $\{P\}\ Q_1\ \{R_1\}\ \Lambda\ \{R_1\}\ Q_2\ \{R\}$

3- إذا تحقق الشرط (1) و(2) سنصل إلى النتيجة النهائية الصحيحة للبرنامج. من خلال حذف تطابقات الجزء الأول والثاني من البرنامج مع ثبات الشرط المسبق للجزء الأول من البرنامج، ونتيجة الجزء الثاني من البرنامج مع أوامر التنفيذ لكليهما. **{R}** $Q_1; Q_2$ **{P}**

مثال تطبيقي:

المثال التالي عبارة عن كود برمجي مكتوب بثلاثية هور و يحتوي على مواصفات البرنامج التي تتمثَّل في الشرط المسبق **{X=x ∧ Y=y}** والشرط اللاحق **{Y=x ∧ X=y}** وعبارات البرنامج التي تتمثَّل في أوامر تنفيذه **R: =X ; X:=Y ; Y:=R** (Gordon,2012,p.25).

{X=x ∧ Y=y} R: =X ; X:=Y ; Y:=R {Y=x ∧ X=y}

المطلوب: الوصول إلى النتيجة النهائية الصحيحة للبرنامج من خلال قاعدة التكوين (التسلسل).

الخطوات: إذا كان لدينا أكثر من أمر تنفيذ للبرنامج كما يلي R: =X ; X:=Y ; Y:=R إذًا لدينا ثلاثة أجزاء للبرنامج (ثلاثة أوامر) سنُقسِّم كل أمر منها على حدة لنصل إلى نتيجة كل جزء من البرنامج.

الجزء الأول من البرنامج:

- طبقًا لتسلسل تنفيذ البرنامج فإن الجزء الأول في تنفيذ البرنامج هو:

{X=x ∧ Y=y} R: =X{Y =x ∧ X=y}

بواسطة بديهية التكليف نستبدل بمتغيرات الشرط اللاحق X،Y قيمته طبقًا للشرط المسبق **{X=x ∧ Y=y}** وأمر التنفيذ **R:=X** لتصبح نتيجة الشرط اللاحق (Gordon,2012,p.25):

{X=x ∧ Y=y} ⟵ **R: =X { R =x ∧ Y=y}** **(1)**

الجزء الثاني من البرنامج:

- نجعل الشرط اللاحق في الجزء الأول من البرنامج كشرط مسبق في الجزء الثاني من البرنامج:

{R =x ∧ Y=y} X: =Y {Y =x ∧ X=y}

- بواسطة بديهية التكليف نستبدل بمتغيرات الشرط اللاحق قيمته تبعًا للشرط المسبق **{R =x ∧ Y=y}** وأمر التنفيذ **X: =Y** لتصبح نتيجة الشرط اللاحق (Gordon,2012,p.25):

{R =x ∧ Y=y} X: =Y {R =x ∧ X=y} ⟵ **(2)**

الجزء الثالث من البرنامج:

- نجعل الشرط اللاحق في الجزء الثاني من البرنامج كشرط مسبق في الجزء الثالث من البرنامج:

{R =x ∧ X=y} Y: =R {Y =x ∧ X=y}

- بواسطة بديهية التكليف نستبدل بمتغيرات الشرط اللاحق قيمته تبعًا للشرط المسبق **{R =x ∧ X=y}** وأمر التنفيذ **Y:=R** لتصبح نتيجة الشرط اللاحق (Gordon,2012,p.25):

{R =x ∧ X=y} Y: =R {Y =x ∧ X=y} ⟵ **(3)**

ب- بواسطة قاعدة التكوين:

- **من خلال (1) و(2) واستخدام قاعدة التكوين (التسلسل)**:(Gordon,2012,p.25)

$$\{X=x \wedge Y=y\}\ R:=X\ \{\sout{R=x \wedge Y=y}\} \wedge \{\sout{R=x \wedge Y=y}\}\ X:=Y\ \{R=x \wedge X=y\}$$

$$\{X=x \wedge Y=y\}\ R:=X;\ X:=Y\ \{R=x \wedge X=y\} \quad (4)$$

- **من خلال (4) و(3) واستخدام قاعدة التكوين (التسلسل)**:

$$\{X=x \wedge Y=y\}\ R:=X;\ X:=Y\ \{\sout{R=x \wedge X=y}\} \wedge \{\sout{R=x \wedge X=y}\}\ Y:=R\ \{Y=x \wedge X=y\}$$

$$\{X=x \wedge Y=y\}\ R:=X;\ X:=Y;\ Y:=R\ \{Y=x \wedge X=y\} \quad (5)$$

3 - قاعدة التكرار Iteration Rule:

التعريف: قاعدة التكرار هي مبدأ الاستنتاج الذي يُستخدَم لاشتقاق التقريرات الصحيحة حول تأثيرات تنفيذ الأمر "**أثناءWhile**" (Goldblatt,1982,p.141). تشير هذه القاعدة إلى القدرة على تنفيذ جزء من البرنامج (Q)بشكل متكرر؛ حتى يصبح الشرط (P) خاطئًا ويُعبّر عن هذا التكرار بواسطة: **while P do Q** Hoare,1969, p578).

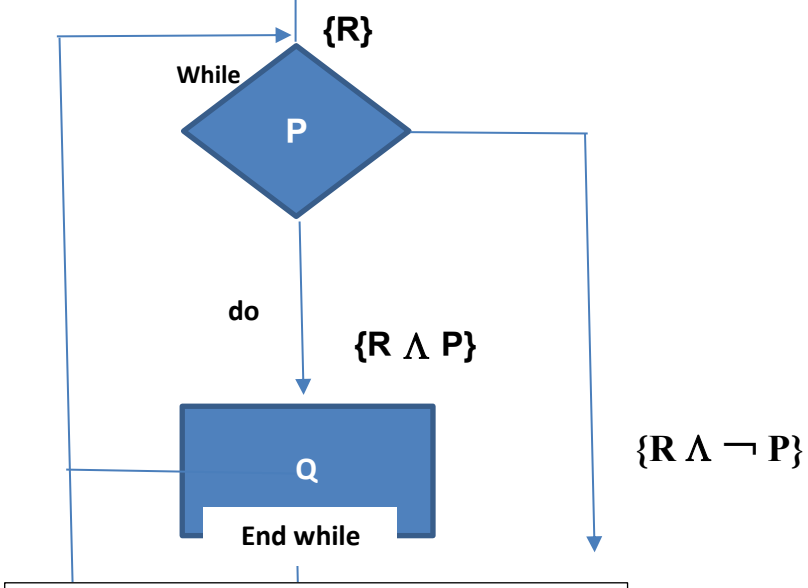
الصياغة الرمزية: عبَّر هور عن قاعدة التكرار من خلال الصياغة الرمزية (Hoare,1969, p578; (Gordon,2012,p.26:

$$\vdash \{R \wedge P\}\ Q\ \{R\}$$

$$\vdash \{R\}\ \textbf{while}\ P\ \textbf{do}\ Q\ \{R \wedge \neg P\}$$

القراءة: أولًا يُختبَر جهاز الحاسوب الشرط P فإذا كان كاذبًا فإن الحلقة التكرارية تكون اكتملت، وخلاف ذلك -أي إذا لم تكن P كاذبة- تُطبق Q وتُختبر P مرة أخرى، ويتكرر هذا الإجراء حتى يثبت أن P كاذبة.(Gordon,2012,p.26). أما R فتسمَّى بالحلقة الثابتة loop invariant؛ لأن هذا التقرير المنطقي عادة ما يكون صحيحًا قبل بدء الحلقة؛ ويبقى صحيحًا بعد كل تنفيذ للحلقة وكذلك حتى نهاية الحلقة (Osho& Ogwueleka,2013,p,69) يقول هور: **"عند افتراض R تقريرًا يكون دائمًا صحيحًا عند استكمال البرنامج Q بشرط أن يكون صحيحًا أيضًا عند البدء، وبالتالي من الواضح أنR تظل صحيحة بعد أي عدد من التكرارات للبرنامج Q** (Hoare,1969, p578)، كما هو موضَّح في الشكل رقم (4).

الشكل رقم (4)
"يوضح قاعدة التكرار"



(Osho& Ogwueleka,2013,p,69)

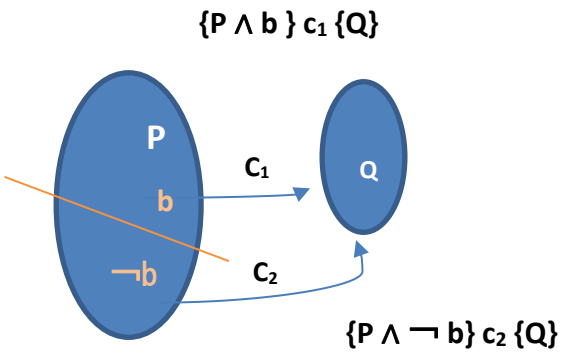
4- القاعدة الشرطية Conditional rule :

يُعبّر عن قاعدة الشرطية في منطق هور من خلال الصياغة التالية (Nipkow& Klein,2014,193):

$$\frac{\{ P \wedge b \}\ c_1\ \{Q\},\ \{ P \wedge \neg b\}\ c_2\ \{Q\}}{\{ P \}\ \text{if } b \text{ then } c_1 \text{ else } c_2 \text{ end } \{ Q \}}$$

القراءة: إذا كانت b صادقة إذا نفّذ c_1 -وخلاف ذلك– أي إذا كانت b كاذبة، نفّذ c_2 ثم قم بإنهاء البرنامج. لذا تُعتبر b قضية نحكم عليها بالصدق أو الكذب طبقًا لشروط القاعدة، التي من خلالها يُنفَّذ البرنامج(Wang,2021,p.33)، كما هو موضَّح في الشكل رقم (5) .

الشكل (5)
"يوضِّح القاعدة الشرطية"



المصدر:(Prasetya,2012.p.18.)

مما سبق نستنتج أن ثلاثية هور تقوم على فكرة **الاشتقاق المنطقي** logical derivation الذي هو عبارة عن "استدلال من مقدمات بعينها، فهو تتابع من قضايا تبدأ من المقدمات وتستمر خلال قضايا أخرى، بحيث تكون كل خطوة هي قضية متضمنة منطقيًّا في القضية السابقة عليها (النويهي، سهام. د.ت، ص73)" إذًا **منطق هور عبارة عن نسق منطقي يسمح لنا بالتحقُّق من صحة البرامج بالاشتقاق المنطقي الدقيق"**(Capretta, 2010,p.2).

رابعًا: نقد وتقييم.

على الرغم من مكانة وأهمية منطق هور باعتباره أول منطق طُبق في مجال من مجالات علوم الحاسوب وهو التحقق من صحة البرامج إلا أن هناك العديد من المشاكل التي تواجه تطبيقه في بعض البرامج ومن أبرز هذه المشكلات:

أولًا: هذا المنطق يمكن تطبيقه فقط على **البرامج البسيطة** التي لا تحتوي على شروط مسبقة كثيرة؛ لأنه إذا كان لدينا برنامج معقد نريد أن نتحقق منه بوضع شروط مسبقة كثيرة، سيكون من الصعب فعل ذلك؛ لأنه سيتطلب وضع قائمة كبيرة بالشروط المسبقة مما يؤدي إلى وجود أخطاء عديدة وصعوبة في حل المشكلة(Frohardt, 2009,pp.3-4).

ثانيًا: منطق هور جيد في التعامل مع البرامج التي تعالج أنواع البيانات البدائية البسيطة مثل الأعداد الصحيحةintegers أو السلاسل strings، لكن براهين هذا المنطق أصبحت أكثر تعقيدًا عند التعامل مع البيانات المهيكلة (المنظَّمة) structured data(*) التي تحتوي على المؤشرات (*) pointers" (O'Hearn,2019,p86).

لذلك يفتقر منطق هور إلى التعبير عن هياكل البيانات المشتركة القابلة للتغيير mutable data structures مثل المصفوفات arrays والمؤشرات(Appel,2014.p.16)، فمثل هذه البيانات تحتاج إلى مواقع كثيرة في ذاكرة الحاسوب، وتتطلب تنظيمًا للمتغيرات ومنطق هور يفتقر إلى هذه الميزة.

ثالثًا: رفض بديهية التكليف؛ لأنها تفرض قيودًا شديدة عند تطبيقها على المؤشرات. ضع في اعتبارك الثلاثية التالية:{y = 1} x:= 2 {y = 1} ينص هذا على أن التخصيص x:= 2 لا يؤثر على قيمة" y إذا كانت قيمتها تساوي " 1"

لا يصلح هذا التخصيص مع العديد من لغات البرمجة الحتمية الشائعة imperative programming languages، مثال ذلك: قد تكون x وy في الواقع ذات اسم مستعارaliased أي أنهما قد يشيران إلى الاسم نفسه، و متداخلان جزئيًّا في نطاق معين لذاكرة الحاسوب. يعني هذا أن يتسبب تغيير قيمة أحد المتغيرات إلى تغيير قيمة المتغيرات الأخرى؛ وهذا بدوره يؤدي إلى صعوبة كبيرة في الكتابة وتصحيح الأخطاء وفهم البرامج (Cartwright& Oppen,1979,p.2 ;Demri& Deters,2015,p.5)

(*) البيانات المهيكلة هي البيانات التي تم تنظيمها في مستودع (قاعدة البيانات)؛ بحيث يمكن جعل عناصرها قابلة للعنونة من أجل معالجة وتحليل أكثر فعالية. هيكل البيانات هو نوع من المستودعات التي تنظّم المعلومات في قاعدة البيانات من أجل الوصول إلى البيانات بطريقة أسهل وأكثر دقة. انظر(Wigmore , 2015).

(*) في مجال البرمجة ومعالجة المعلومات، هو متغير يحتوي على موقع الذاكرةmemory location (العنوان) لبعض البيانات بدلًا من البيانات نفسها. وتختلف قيمته في ذاكرة الحاسوب. انظر:(Press, 2002.p.410).

رابعًا: من ناحية أخرى لاحظ أنه في حالة استدعاء العميل لدالة ما دون استيفاء شرطها المسبق، يمكن للدالة أن تتصرَّف بأي شكل من الأشكال ولا تزال صحيحة. لذلك من المفترض أن تكون الدالة قوية في مواجهة الأخطاء، كما يجب أن يكون الشرط المسبق يشتمل على إمكان الإدخال الخاطئ، ويجب أن يصف الشرط اللاحق ما يجب أن يحدث في حالة هذا الإدخال الخاطئ (Aldrich,2013.p.1).

لذلك **اتفق علماء الحاسوب على حقيقة أنه من الأفضل التحقُّق من برامج السلامة الحرجة، والأجهزة في الوقت الحاضر والتأكُّد من صحتها وخلوِها من الأخطاء قبل تشغيلها، بواسطة التحقُّق الصوري الآلي automated formal verification، من خلال الأنظمة الصورية التي ظهرت لأول مرة مع منطق هور، ولكن يتطلب الأمر تطوير هذا المنطق ليستوعب حل مثل هذه المشكلات** (Cousot& Radhia,2010,p.1).

بناء على ما سبق سعي مجموعة من العلماء لمعالجة بعض قيود منطق هور في العديد من البحوث. على سبيل المثال المقالة البحثية لعالم الحاسوب والمنطق أوهيرن O'Hearn حول" منطق الانفصال" separation logic التي ركَّزت على وضع أسس منطق جديد؛ لتطبيقه في التحقُّق من صحة البرامج التي تتعامل مع المؤشرات.

إن إجراء المزيد من الأبحاث حول دور المنطق في التحقُّق من البرامج قد يسلِّط الضوء على المشكلات المختلفة المحتمَل حدوثها في البرامج، والتي يسعى علماء الحاسوب، والمنطق، والمبرمجون، والمطوّرون إلى حلها بواسطة الربط بين الجانبين، المنطقي، والتقني، وهذا بدوره يساهم في تطوير البرامج وزيادة كفاءتها في المستقبل.

الخاتمة

لقد أظهر البحث الحالي العديد من النتائج تُوجز فيما يلي:

- **لقد مر منطق هور بمرحلتين أساسيتين:**

– مرحلة الظهور متمثِّلة في أعمال **"فلويد"** عام1967 في مقالته **"تحديد المعنى للبرامج"**؛ حيث استخدم التقريرات لإثبات صحة البرامج على المخططات الانسيابية flowcharts.

– مرحلة التطور متمثِّلة في أعمال **"هور"** عام 1969 في مقالته **"الأسس البديهية لبرمجة الحاسوب"**؛ لأن البديهيات لديه هي أساس النسق الاستنباطي لإثبات صحة البرامج النصية من خلال ثلاثية هور، ولا يخلو برنامج لديه من وجود بديهية.

- استمد هور أساسيات -منطقه- من مصدرين أساسيين:
- تطبيق تورين لصحة التقريرات لضمان صحة برامج المخططات الانسيابية.

– الطريقة الصورية التي وضعها فلويد لصحة برامج المخططات الانسيابية من خلال التقريرات الاستقرائية.

– الأسس المنطقية لمنطق هور مستقاة من منطق الدرجة الأولى في استخدام **المحمولات** و**الأسوار الكمية** و**الحدود (ثوابت ومتغيرات)**، ومنطق القضايا **في استخدام الروابط المنطقية.**

- الصياغة الرمزية الأساسية لمنطق هور هي صياغة **"ثلاثية هور"**، وعلى أساسها قدم هور تعريفًا للصياغات الخاصة بالبديهيات وقواعد الاستدلال؛ للتأكد من صحة هذه الثلاثية وإمكان تطبيقها على البرامج.
- تتكوَّن ثلاثية هور من:

– **التقريرات المنطقية**: تتمثَّل في الشروط المسبقة والشروط اللاحقة.

– **أمر البرنامج**: هو الأمر الذي يتوقف تنفيذه على صدق الشروط المسبقة.

- الشروط المسبقة والشروط اللاحقة **قضايا منطقية** توضِّح بعض خصائص المتغيرات المستخدَمة في البرنامج، ونحكم عليها بالصدق أو الكذب في حين أن الثلاثية نحكم عليها بالصحة أو عدم الصحة.
- يتم تحديد صحة الثلاثية من عدمها من خلال استخدام النسق الاستنباطي ، الذي يحتوي على:

– البديهيات (بديهية التخطي- بديهية التكليف).

– استخدم هور روابط المنطق الكلاسيكي (**العطف** و**اللزوم** و**النفي)** في تعريف قواعد الاستدلال الخاصة بنسقه (قاعدة التكوين – قاعدة النتيجة – قاعدة التكرار- قاعدة الشرطية).

- يُعتبر الاستنباط في منطق هور **استنباطًا طبيعيًّا؛** وذلك لأن منطق هور له بعض سمات الاستنباط الطبيعي، وهي:

– لا يقتصر على البديهيات فقط، ولكن يستخدم منطق هور مجموعة من البديهيات وأربع قواعد للاستدلال.

– يُستخدم منطق هور لإثبات صحة البرامج.

– المقدمات في منطق هور سواء أكانت بديهيات أم مجموعة من الحُجج تُقدَّم بحرية.

– يستخدم منطق هور قواعد الاستدلال ذات الطابع البسيط والذي يوضح كيفية تكوين الصيغ وتحليلها.

– تميل البراهين في هذا النسق إلى أن تكون أسهل وأقصر منها في أنظمة البديهيات الغامضة.

- يُعتبر "منطق هور" هو **"حساب هور"**؛ لأن صحة أو عدم صحة ثلاثية هور يتم معرفتها من خلال قيم صدق الشروط المسبقة والشروط اللاحقة لثلاثية هور، وذلك باستخدام قواعد النسق الاستنباطي.
- المنطق يساهم في حل العديد من مشكلات علوم الحاسوب، ولكن لكل منطق آلية وأسس لحل المشكلة تبعًا لنوع المشكلة والهدف الذي يسعى إليه المنطق، فمنطق هور يندرج تحت منطق البرامج؛ للتحقق من صحة برامج الأنظمة الحرجة البسيطة؛ من خلال تحديد **سلوك البرامج** والاستدلال عليها بواسطة **ثلاثية الصحة الجزئية.**
- على الرغم من وجود تداخل بين التحقق من البرامج واختبار البرنامج، وإثبات صحة البرنامج، ولكن هناك تباين بينهم

- **التحقق من البرامج:** هي محاولة لإثبات أن برامج الحاسوب صحيحة والبرنامج يكون صحيحًا إذا كان يتصرف وفقًا لمواصفاته.

- **اختبار البرنامج:** يسعى لإظهار أن قيم مدخلات معينة تنتج قيم مخرجات مقبولة.

_ **إثبات صحة البرامج**: يستخدم نسقًا منطقيًّا لإثبات إذا كانت متغيرات المدخلات تفي ببعض المحمولات أو الخصائص المحددة، فإن متغيرات المخرجات الناتجة عن تنفيذ البرنامج تفي بخصائص محددة.

- هناك اختلاف بين أدوات المبرمج الجيد والمبرمج غير الجيد، ونتائج كلٍّ منهما في برمجة الحاسوب:
  - **المبرمج الجيد** أداته المنطق ويترتب على ذلك قلة الأخطاء، سرعة حل المشكلة، توفير الوقت والجهد.
  - **المبرمج غير الجيد** أداته التجربة والخطأ ويترتب على ذلك كثرة الأخطاء، بطيء في حل المشكلة، استهلاك المزيد من الوقت لحل المشكلات.
- **هناك طريقتان للتحقُّق من صحة البرامج:**

الطريقة التقليدية: تتمثَّل في فكرة "التجربة والخطأ" ومقياس صحة البرنامج هو اكتشاف الأخطاء وإزالتها.

الطريقة الجديدة: تتمثَّل في "الأسلوب المنطقي" ومقياس صحة البرنامج هي إثبات عدم وجود أخطاء؛ من خلال استخدام البديهيات وقواعد الاستدلال الصحيحة.

- من أهم عيوب منطق هور أن تطبيقه يتعلق **"بالبساطة"**:
  - يقتصر تطبيقه على البرامج البسيطة؛ لذلك لا يمكن تطبيقه على البرامج المعقدة.
  - يتعلق بأنواع بسيطة من البيانات مثل (الأعداد – الحروف)؛ لذلك يفتقر تطبيقه على البيانات المهيكلة مثل (المصفوفات والمؤشرات).

التوصيات

وفقًا لموضوع البحث، والنتائج التي تم التوصَّل إليها، توصي الباحثة بـ:

1- تفعيل دور عالمنا العربي في تطوير المنطق، وتطبيقه في مجالات علوم الحاسوب عن طريق: المشاركة بين متخصصي المنطق، ومتخصصي علوم الحاسوب في عالمنا العربي لنشر أبحاث حول تطبيقات المنطق في علوم الحاسوب.

2- تقديم دراسة مستفيضة حول منطق هور وتطبيقاته في مجال التحقق من صحة البرامج

3- التطرق لموضوعات بحثية جديدة مثل:
   - صحة برامج الحاسوب بين منطق الدرجة الأولي ومنطق هور.
   - كيف يساهم المنطق في تطوير مهارات دراسي علوم الحاسوب.

المصادر والمراجع

النويهي، سهام (بدون تاريخ) ***أسس المنطق الرياضي*** .كلية البنات .جامعة عين شمس.

عبد الجواد، أحمد.(2022). ***مفهوم الاستلزام عند نيلسون*** .مجلة البحث العلمي في الآداب(2)*23* .

Adrion, W. R., Branstad, M. A., & Cherniavsky, J. C. (1982). ***Validation, verification, and testing of computer software***. ACM Computing Surveys (CSUR), 14(2),p.161.

Aldrich,Jonathan,(2013), ***Lecture Notes: Hoare Logic***. Lecture 3.

Almeida, J. B., Frade, M. J., Pinto, J. S., & De Sousa, S. M. (2011). ***Rigorous software development: an introduction to program verification***. (Vol. 1). London: Springer.

Amarendra, K., & Rao, A. V. (2011). ***Safety critical systems analysis. Global journal of computer science and technology***, 11(21).

Appel, A. W. (2014). ***Program logics for certified compilers***. Cambridge University Press.

Apt, K. R., & Olderog, E. R. (2019). ***Fifty years of Hoare's logic***. Formal Aspects of Computing, 31.

Attie,Paul(2001).***Axiomatic Semantics and Program Verification*** ,College of Computer Science Northeastern University.

Backhouse, Roland (2002).***Hoare Triples and the Assignment Axiom.***(n.p)**.**

Barthe,Gilles (2020) .***An introduction to relational program verification***. Vol.xx .No..xx The essence of knowledge.Boston.

Ben-Ari, M. (2012). ***Mathematical logic for computer science***. Springer Science & Business Media.

Bernot, G., Comet, J. P., Khalis, Z., Richard, A., & Roux, O. (2019). ***A genetically modified Hoare logic***. Theoretical Computer Science, 765.

Capretta,Venanzio (2010).***Hoare Logic "G52DOA - Derivation of Algorithms***, School of Computer Science. University of Nottingham.

Cartwright, R., & Oppen, D.(1979) ***The logic of aliasing***. Stanford Verification Group.No.12

Chowdhary, K. R. (2020). ***Fundamentals of artificial intelligence*** .New Delhi.Springer India.

Cousot, P. (1990). **Methods and logics for proving programs**. In Formal Models and Semantics .Elsevier.

_____ (2010). ***A gentle introduction to formal verification of computer systems by abstract interpretation***. In Logics and Languages for Reliability and Security. IOS Press.

Corcoran, John &Hamid, Idris (2022).**Schema**", The Stanford Encyclopedia of Philosophy. retrieved from https://plato.stanford.edu/entries/schema/#WhatScher

Crăciun,Adrian (2017). ***Logic for computer science*** . A Practical Introduction for Computer Science Students.

Demri, S., & Deters, M. (2015). ***Separation logics and modalities: a survey***. *Journal of Applied Non-Classical Logics*, *25*(1).

Deransart, P., Hermenegildo, M. V., & Maluszynski, J. (Eds.). (2000). ***Analysis and visualization tools for constraint programming: constraint debugging*** (No. 1870). Springer Science & Business Media.

Dijkstra, E. W. (1972). ***The humble programmer***. Communications of the ACM, 15(10).

Drabent, W., & Małuszyński, J. (1987, March). I***nductive assertion method for logic programs. In International Joint Conference on Theory and Practice of Software Development*** . Berlin, Heidelberg: Springer Berlin Heidelberg.

D'Souza, Deepak Raghavan, K. V. (2012*).* ***Hoare Logic***. Indian Institute of Science, Bangalore.

Feng, X., Müller-Olm, M., & Yang, Z. (Eds.). (2018). ***Dependable Software Engineering. Theories, Tools, and Applications****: 4th International Symposium, SETTA 2018, Beijing, China, September 4-6, 2018, Proceedings* (Vol. 10998). Springer.

Floyd,Robert .(1967). **Assigning Meanings to Programs**, American Mathematical Society. U.S.A.

Frohardt, R. (2009). ***A Brief Introduction to Separation Logic***.(n.p)**.**

Geeksforgeeks (2021). **Difference between Routine and Process**. retrieved from https://www.geeksforgeeks.org/difference-between-routine-and-process/ on 10-6-2022 .5;13 pm.

Gore,Rajeev (2016).***Hoare Logic. Formal Methods for Software Engineering***. Australian National University.

Goldblatt, Robert (1982).***The Semantics of Hoare's Iteration Rule*.** Studia logica, Victoria University.

Gordon, Michael (1988). ***Mechanizing programming logics in higher order logic***. N(145).Technical Report. University of Cambridge.

Gordon, Mike (2012). ***Background reading on Hoare logic***. *Lecture Notes, April*.

Gries, David. (Ed.). (1978). ***Programming methodology*** .[1]t edition .Springer-Verlag New York.

Harmim, Dominik, Vladimır Marcin, and Ondrej Pavela. (2019)**.*Scalable static analysis using facebook infer***. Brono Faculty. University of information of technology.

Hoare,Tony (1972).***A Note on the FOR Statement*.**BIT.

_____ (1969). ***An axiomatic basis for computer programming***. *Communications of the ACM*, *12*(10).

Hodges, Andrew (2014).***Alan Turing: The Enigma***, United Kingdom, London.

Hosch,William.(2009). ***Tony Hoare*** .**retrieved from** https://www.britannica.com/biography/Tony-Hoare/additional-info#history) .

Jones,Cliff (1984 April).***An early program proof by Alan Turing***,. *IEEE Annals of the History of Computing*, *6*(02).

_____ (2010*).****Reflections on the Work of C.A.R. Hoare****,* Springer London Dordrecht Heidelberg NewYork.

Hodges, Andrew (2014).***Alan Turing: The Enigma***, United Kingdom, London.

Knuth, D. E. (2003). ***Robert w floyd, in memoriam***. *ACM SIGACT News*, *34*(4).

Lee, J.A.N(Ed.). (1995**).** ***International biographical dictionary of computer pioneers***. IEEE computer Society press.

Leroy, Kenneth & Braunchweig, Dave.(2018) . ***Programming Fundamentals*** ,2nd Edition. creative commons Attribution.

Manna, Z., & Waldinger, R. (1978). ***The logic of computer programming***. *IEEE transactions on Software Engineering*, (3).

Müller, P., & Schaefer, I. (2018). ***Principled Software Development***. Springer International Publishing.

Naumann,David .(2009).***Theory for Software Verification***, ACM Journal Name, Vol. V, No. N, January.

Nielson, Hanne(1984). ***Hoare logics for run - time analysis of programs***, University of Edinburgh.

Nipkow,Tobias.Klein, Gerwin (2014).***Concrete Semantics***, Springer International Publishing Switzerland.

O'Hearn, Peter (2019). ***Separation logic***. *Communications of the ACM*, *62*(2), 86-95.

Osho,Lauretta. Ogwueleka,Francisca.(2013).**Axiomatic Basis for Computer Programming**," Universal Journal of Computational Mathematics, 1(3).

Pharabod,Jean (2017) .**Hoare logic (Formalising the semantics of Hoare logic)**, Lecture 2, CST Part II, University of Cambridge.

Poroor, Jayaraj (2021). ***Natural Hoare Logic: Towards formal verification of programs from logical forms of natural language specifications***. *arXiv preprint arXiv:2103.05779*.

Prasetya ,Wishnu (2012*).* ***Introduction to Programming Logic.*** Utrecht University.

Puebla, German. Bueno, Francisco (2000).***An Assertion Language for Constraint logic programs****,* Springer- Verlag Berlin Heidelberg.

Press, Microsoft. (2002). ***Microsoft computer dictionary***. 5th ed. **.**(n.p**).**

Reese, Richard (2013).***Understanding and Using C Pointers***, First Edition. y O'Reilly Media.

Rose, B. W. (1994). **Fatal dose**: ***Radiation deaths linked to AECL computer errors***. *Canadian Coalition for Nuclear Responsibility*.

Schreiner, Wolfgang (2021). ***Thinking Programs***. Springer International Publishing.

Svendsen,Kasper (2016).**Hoare logic and Model Checking**, University Cambridges -Part II-2016.

Swenker, Remco (2012). ***Translating Hoare Logic to SMT, the Making of a Proof Checker***. University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science. June 25th.

Tarkoma,Sasu (2003).***Specification Languages and Their Use (Case: Asml)***, Seminar on Generative Programming University of Helsinki, Department of Computer Science, Spring.

Techopedia (n.d).***Assignment***. retrieved from https://www.techopedia.com/definition/17746/assignment-programming.on 29-5-2022 .5:52

Turing,Alan (1949).***On checking a large routine***. Report of a conference on high-speed automatic calculating machines, University Mathematics Laboratory, Cambridge.

Wang,Yi (2021).***an implementation of separation logic in coq***, (Master of Science) Faculty of Mathematics and Science, Brock University St.Catharines, Ontario.

Wiedijk, F. A (n.d).***large routine***. *Liber Amicorum voor/for Jan A. Bergstra*.

Wigmore ,Ivy (2015) .***structured data***, Received from **https://www.techtarget.com/whatis/definition/structured-data** on 9-8-2022 . 5:23 am.