Volume 1, Issue 2, December 2021 - ISSN 2735-5985

مركز الأكاديمية للنشر
ACADEMY Publishing Center

Journal of Advances in Computing and Engineering

Academy Publishing Center
Journal of Advances in Computing and Engineering (ACE) First edition 2021

Arab Academy for Science, Technology, and Maritime Transport, AASTMT Abu Kir Campus, Alexandria, EGYPT
P.O. Box: Miami 1029
Tel: (+203) 5622366/88 – EXT 1069 and (+203) 5611818
Fax: (+203) 5611818
Web Site: http://apc.aast.edu

Volume 1, Issue 2, December 2021
ISSN: 2735-5977 Print Version
ISSN: 2735-5985 Online Version

ACE
Journal of Advances in Computing and Engineering

Journal of Advances in Computing and Engineering (ACE) is a peer-reviewed, open access, interdisciplinary journal with two issues yearly. The focus of the journal is on theories, methods, and applications in computing and engineering. The journal covers all areas of computing, engineering, and technology, including interdisciplinary topics.
All submitted articles should report original, previously unpublished research results, experimental or theoretical. Articles submitted to the journal should meet these criteria and must not be under consideration for publication elsewhere. The manuscript should follow the style specified by the journal.
ACE also publishes survey and review articles in the scope. The scope of ACE includes but not limited to the following topics: Computer science theory; Algorithms; Intelligent computing; Bioinformatics; Health informatics; Deep learning; Networks; Wireless communication systems; Signal processing; Robotics; Optical design engineering; Sensors.

The journal is open-access with a liberal Creative Commons Attribution-NonCommercial 4.0 International License which preserves the copyrights of published materials to the authors and protects it from unauthorized commercial use.
The ACE journal does not collect Articles Processing Charges (APCs) or submission fees, and is free of charge for authors and readers, and operates an online submission with the peer review system allowing authors to submit articles online and track their progress via its web interface.
The journal is financially supported by the Arab Academy for Science, Technology, and Maritime Transport AASTMT in order to maintain quality open-access source of research papers on Computing and Engineering.

ACE has an outstanding editorial and advisory board of eminent scientists, researchers and engineers who contribute and enrich the journal with their vast experience in different fields of interest to the journal.

Advisory Board

Administrative Committee

Journal Manager

Copy Editor

Layout Editor and Proof-reader

IT Manager

TABLE OF CONTENTS

http://dx.doi.org/10.21622/ACE.2021.01.2.033

Visible Light Communications Illuminates the Future

Moustafa H. Aly

Electronics and Communications Engineering, College of Engineering and Technology,
Arab Academy for Science, Technology and Maritime Transport, Alexandria 1029, Egypt

Email: {mosaly@aast.edu}

The use of Internet, video and audio calls at a very low cost is now common for all people around the world. This implies a tremendous carrying capacity for the carrier wave. The credit gets back to the use of light as a carrier wave, leading to Optical Communications.
The use of visible optical carrier waves or light for communication has been common for many years. Simple systems such as signal fires, reflecting mirrors and, more recently, signaling lamps have provided successful, information transfer. Moreover, as early as 1880, Alexander Graham Bell reported the transmission of speech using a light beam.

Depending on their wavelengths, these electromagnetic carriers can be transmitted over considerable distances, but are limited in the amount of information they can convey by their frequencies. In this context, it may also be noted that communication at optical frequencies offers an increase in the potential usable bandwidth by a factor of around 104 over high-frequency microwave transmission. A renewed interest in optical communication was stimulated in the early 1960s with the invention of the laser, considering it as a single frequency source. This device provided a powerful coherent light source, together with the possibility of modulation at high frequency.

However, the previously mentioned constraints of light transmission in the atmosphere tended to restrict these systems to short-distance applications. The proposals for optical communication via optical fibers fabricated from glass to avoid degradation of the optical signal by the atmosphere were made in 1966 by Hockham. Such systems were viewed as a replacement for coaxial cable or carrier transmission systems [1].

Many studies were conducted to depict the best carrier wavelengths to be used in optical fiber communications. Two windows were chosen: 1.3 μm and 1.55 μm, where single mode fibers achieved minimum loss and maximum transmission bit rates. Millions of kilometers of optical fiber cables, either underground or undersea, are responsible of carrying different data around the globe.

In addition to optical fiber communications, the wireless optical communications shared the revolution of optical communications in both outdoor and indoor applications. This was achieved using visible light, leading to Visible Light Communications (VLC) [2, 3]. The VLC system can also be referred as LiFi (Light Fidelity), analogous to the term WiFi. In 2011, Harald Haas was the first to coin the term Light Fidelity (Li-Fi). Li-Fi is a high speed bi-directional fully connected, visible light wireless communication system. In the areas that are sensitive to electromagnetic radiation (such as aircrafts), Li-Fi can be a better solution. A Li-Fi also lends support to the Internet of Things (IoT). VLC uses wavelengths between 380 nm to 750 nm (i.e. 430 THz to 790 THz) for communication, Fig. 1.

Volume 1, Issue 2, December 2021 - ISSN 2735-5985
http://dx.doi.org/10.21622/ACE.2021.01.2.033

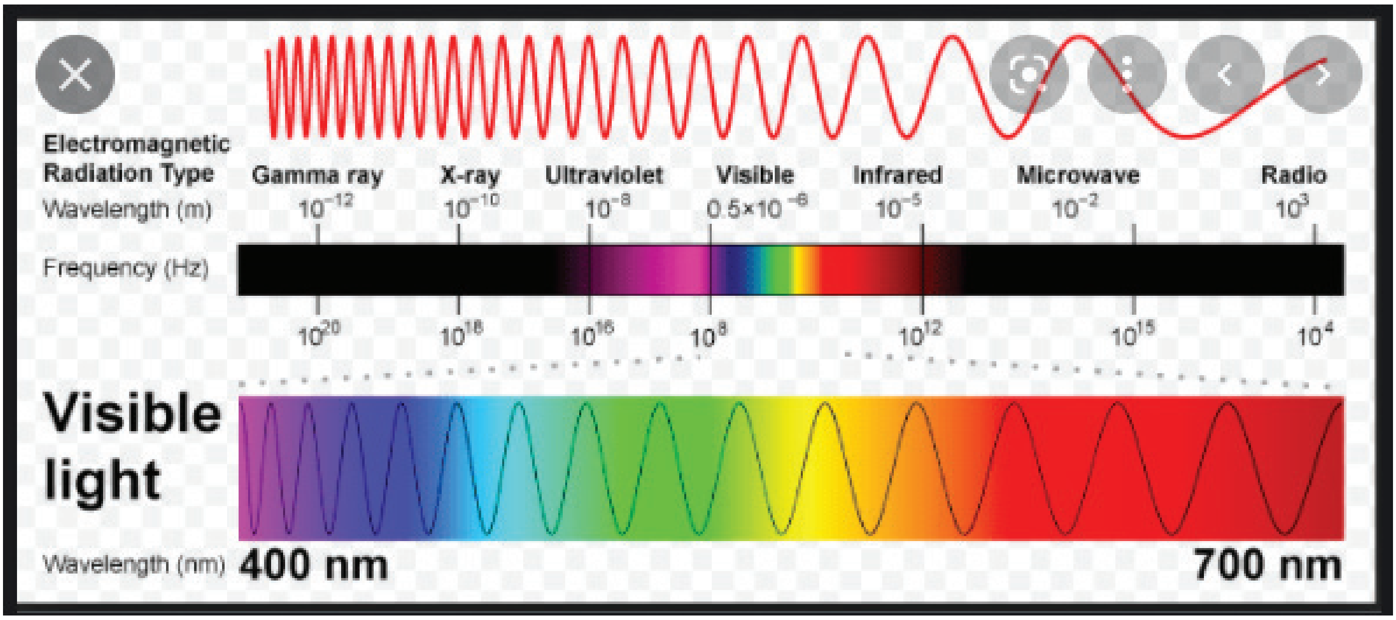
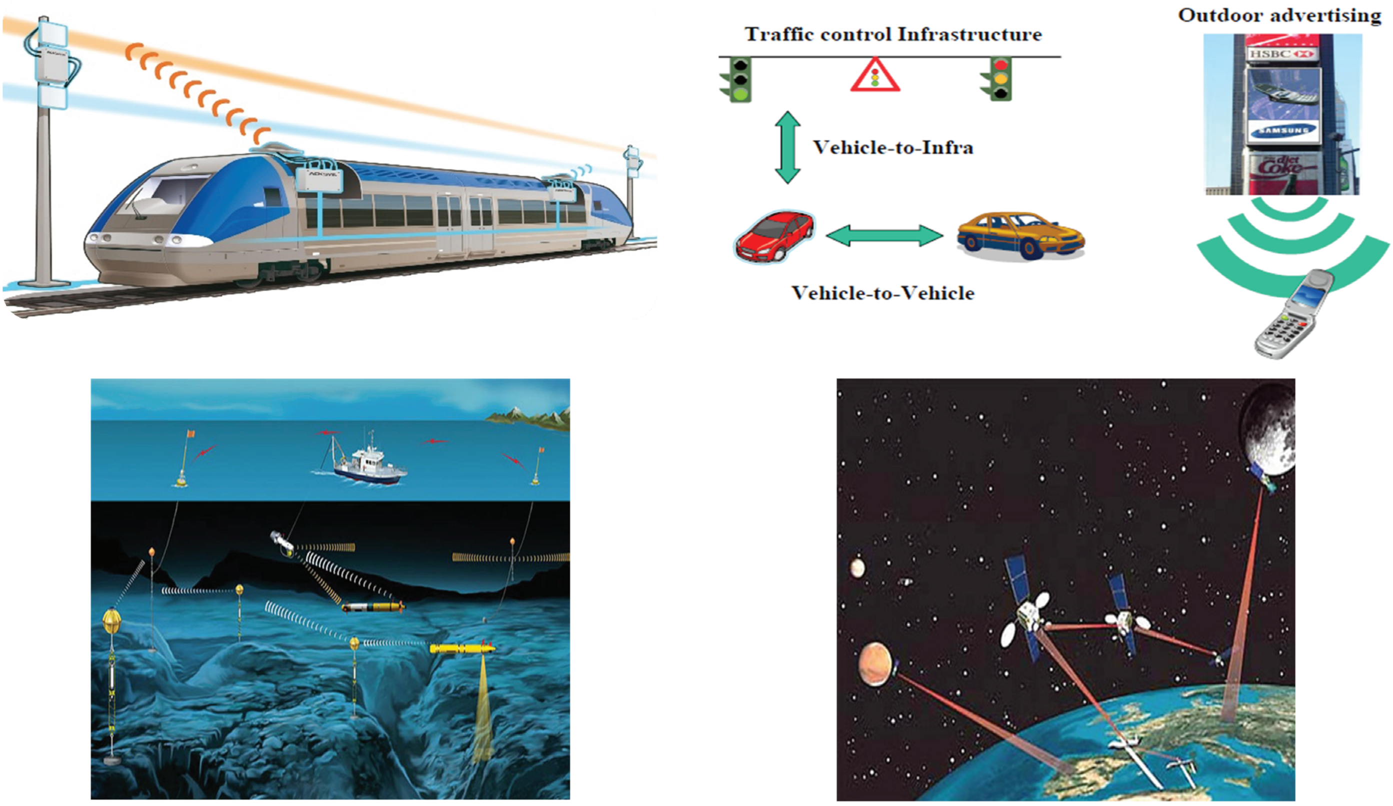


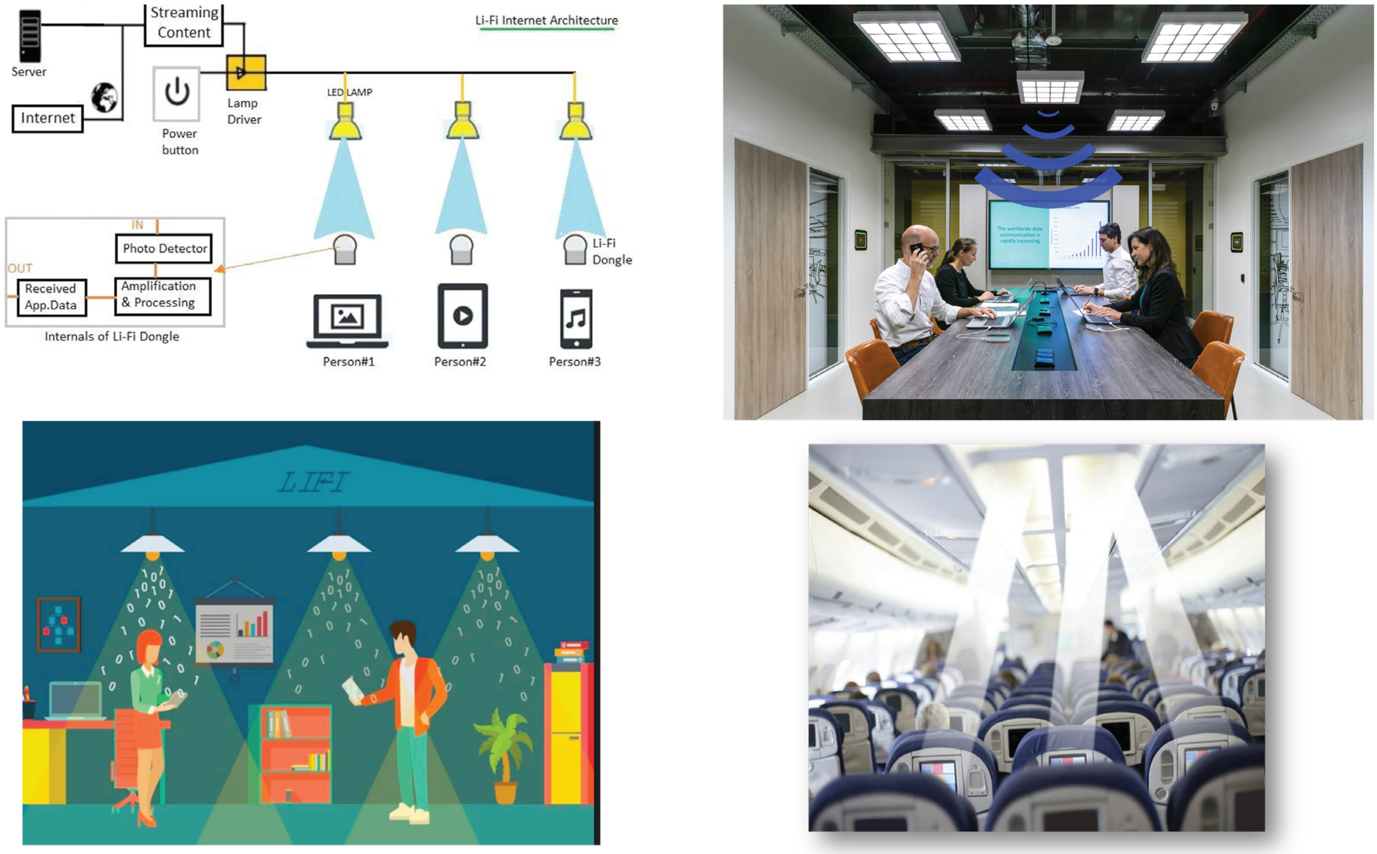
Figure 1 Light spectrum [1].

Some of VLC applications include under water communication, vehicle to vehicle communication, indoor broadcast system for internet use and inside airplane communication. In indoor applications, the light emitting diodes (LEDs) are used for dual purposes: illumination and communications [4]. Examples of these applications are illustrated in Figure 2.



(a)

http://dx.doi.org/10.21622/ACE.2021.01.2.033



(b)

Figure 2 (a) VLC outdoor applications [2, 4]. (b) VLC indoor applications [3].

VLC has many advantages including:

- It supports larger bandwidth and hence overcomes bandwidth limitation of RF communication.
- It provides secured communication unlike RF communication.
- The VLC source is used for both illumination and communication. Hence, VLC is power efficient system.
- VLC is light based communication. Hence, it is not affected by electromagnetic radiation from RF systems, leading to safety from health risks. It is easy to install.
- No license is required.

The VLC still has some drawbacks and challenges that have to be solved. This includes:

- VLC based communication has interference issues from other ambient light sources.
- VLC communication supports short coverage range. So, there are challenges to integrate VLC with WiFi systems.
- Atmospheric effects.
- A fundamental challenge in VLC systems is that they require the light ON all time. Then, how to communicate when the lights appears "OFF"? Here, it appears a new and a promising expression: "Dark Light", where visible light can be used to transmit data even when the light appears dark or off. The VLC minimum illumination to receive power is investigated to enable users to receive data when the light is "OFF" [5].

REFERENCES

[1] J.M. Senior, Optical Fiber Communications: Principles and Practice, 3rd ed., Pearson Education Limited, London, UK, 2009.

[2] Latif Ullah Khan, "Visible light communication: Applications, architecture, standardization and research challenges," Digital Communications and Networks, vol. 3, no. 2, pp. 78-88, 2017.

[3] G. Tiwari, R. Chandra, and S. Chauhan, "A review on inter-satellite links free space optical communication," Indian Journal of Science and Technology, vol. 13, no. 6, pp. 712 – 724, 2020.

[4] G.A. Mapunda, R. Ramogomana, L. Marata, B. Basutli, A.S. Khan and J.M. Chuma, "Indoor visible light communication: A tutorial and survey," Wireless Communications and Mobile Computing, vol. 2020, Article 8881305, pp. 1-46, 2020.

[5] Marwa M. El-Gamal, Maheswar R, Heba A. Fayed, Moustafa H. Aly, Nour Eldin H. Ismail, and Amr Mokhtar, "Dark light visible light communication positioning system with received signal strength technique," Optical and Quantum Electronics, vol. 53:42(1-16), 2021.

BIOGRAPHY

__Moustafa H. Aly__ was born in 1953, Alexandria, Egypt. He received his B.Sc., M.Sc. and Ph.D. from Faculty of Engineering, Alexandria University, Alexandria, Egypt, respectively, in 1976, 1983 and 1987. He is a professor of Optical Communications, Electronics and Communications Engineering Department, College of Engineering and Technology, Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt. He was a co-supervisor of 150 M.Sc. and Ph.D. students and published 330 journal and conference papers. His research area includes Optical Communications, Optical Amplifiers, Free Space Optics, Visible Light Communications, and Optical Networks.

http://dx.doi.org/10.21622/ACE.2021.01.2.037

The 'New Normal' in Higher Education: Innovative teaching and learning technologies and practices during a crisis

Nicholas Blessing Mavengere[1], Shepard Pondiwa[2], Tinashe Charles Matiyenga[3], Francis Manzira[4] and Alfred Mutanga[5]

Bournemouth University, United Kingdom[1]. Midlands State University, Zimbabwe[2]
National University of Science and Technology, Zimbabwe[3]. University of Venda, South Africa[4]. University of Limpopo[5]

Email: {nmavengere@bournemouth.ac.uk> <pondiwas@msu.ac.zw> <tinashe.matiyenga@nust.ac.zw> <Francis.Manzira@univen.ac.za> <alfred.mutanga@ul.ac.za}

Received on, 15 July 2021 - Accepted on, 30 August 2021 - Published on, 10 November 2021

ABSTRACT

Purpose

The purpose of this paper is to analyze the effectiveness of innovative technologies and practices adopted to enhance teaching and learning in higher education settings during a crisis e.g. the COVID-19 pandemic in developing countries. We seek to highlight the need for digital and technological transformation in the teaching and learning practices in response to crisis for adaptation to the 'new normal'.

Design /Methodology approach

Using a qualitative research approach, data was collected through an online questionnaire and online interviews with lecturers who conducted teaching during the COVID-19 pandemic in developing countries. The countries from where data was collected are Egypt, Zimbabwe, South Africa, Tanzania, Namibia and Botswana. The aim was to highlight transformational processes required for to remote learning during the COVID-19 pandemic. In doing so, we note the adaptable approaches that are useful in times of crisis to promote learning.

Findings, contributions and implications

The results highlight measures adopted to promote remote learning during a crisis, such as, COVID-19. The contributions include a conceptual model for innovative technologies and practices to the rapid shift to remote learning and ensuring qualities of learning. In developing countries' context, making use of existing infrastructure, system and organizations is necessary to promote learning in higher education.

1. INTRODUCTION

There have been changes in the way teaching and learning is conducted in tertiary institutions throughout the world in recent years and challenges have been observed (Martinez, 2020; Rapanta et al, 2020; UNESCO, 2020)[1,2,3]. Most of these transformations have been in response to a crisis. Some institutions have transformed their programmes in response to natural disasters while others' transformation has been in response specifically to the COVID-19 pandemic (Dhawan, 2020; Mishra et al, 2020, Wu, 2020) [4,5,6]. The purpose of this paper is to analyze innovative technologies and practices that have been adopted to enhance teaching and learning in higher education settings during a crisis e.g. the COVID-19 pandemic in developing countries. The paper also discusses the impact of the 'new normal' on active learning, collaborative learning, constructive learning, contextual learning, reflective learning and transfer learning. In addition, the paper highlights the need for technological transformation in the teaching and learning practices in response to crisis for adaptation to the 'new normal'. The "new normal" in this research refers to the state to which higher education institutes settle following a crisis when this differs from the situation that prevailed prior to the start of the crisis, in this case COVID-19 pandemic.

There is literature, already in the higher education domain, that shows that there have been some efforts to transform teaching and learning in times of a crisis, for instance, ["Author" 2021; Motala and Menon, 2020; UNESCO, 2020] [3,7,8]. The outbreak of COVID-19 occasioned the massive adoption of virtual learning platforms in many institutions of higher learning. It has become inevitable, in the current COVID-19 era, for educators to integrate ICT in their teaching and rapidly replace traditional teaching methods with modern ones which are ICT based. The provision of education using ICT and mobile communication platforms has become a common practice at most institutions particularly in developing countries. We are motivated to investigate this ICT adoption and application in times of crisis by higher education as summed up by the research question: how can innovative technologies and practices be applied in times of crisis, such as a pandemic in developing countries to ensure qualities of learning?

2. THEORETICAL BACKGROUND

2.1 Qualities of learning

We should seek to achieve qualities of learning in higher education (Baran 2011; Wang and Torrisi-Steele 2015; "Author" 2018, Wu 2020) [6,9,10,11]. Technology could be adopted in efforts to foster qualities of learning. Interest in active learning is increasing, many instructors are trying out new procedures and report them to their fellow instructors (Hartikainen et al., 2019; Motala and Menon, 2020; UNESCO, 2020; Wu, 2020) [3,6,8,12]. In this research we investigate the use of technology to enable learning and also maintain or further improve the quality of learning during critical times. Ruokamo and Pohjolainen (1998, p. 293) [13] suggest the following qualities of learning;

1. Active - Learning process is active, and learners are engaged in mindful processing and they are responsible for the result.
2. Constructive - Knowledge is constructed from previous knowledge by learners.
3. Collaborative - Learners work in co-operation with each other and exploiting each other's skills in building new knowledge
4. Intentional - A cognitive objective is set and learners try actively and willingly to achieve it.
5. Contextual – Learning is based on problem-based real life examples or they are introduced through case-based i.e. meaningful real-world tasks.
6. Transfer – Learners' ability to transfer learning from different situations and contexts.
7. Reflective – Learners' ability to sum up what they have learned including the process and outcomes.

2.2 Community of Inquiry

Learning conducted through the online environment and platforms should still attain qualities of learning. To attain this, the community of inquiry model defines a good virtual learning environment (Garrison and Arbaugh, 2007) [14]. This is done through three major components namely cognitive presence, social presence and teaching presence as illustrated in Figure 1. Lowenthal and Lowenthal (2010, p1) [15] considered "social presence as a theory that explains the ability of people to present themselves as 'real people' through a communication medium. Most studies on social presence focus on how students present themselves and/or are perceived as 'real' people online". Garrison et al. (2000) [16] suggest emotional expression, open communication and group cohesion as social presence indicators. Cognitive presence is the ability to construct and confirm meaning through sustained reflection (Anderson, Rourke, Garrison, & Archer, 2001). Fiock (2020) [17,18] emphasizes the value of learners' reflection to increase cognitive presence. Teaching presence is the design, facilitation and direction of cognitive and social presence to support learning (Garrison and Arbaugh, 2007; Garrison et al., 2000) [14,16].

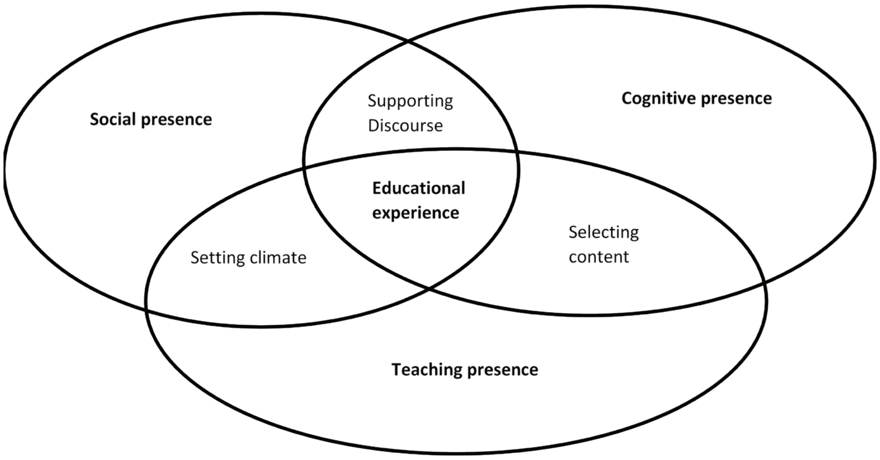
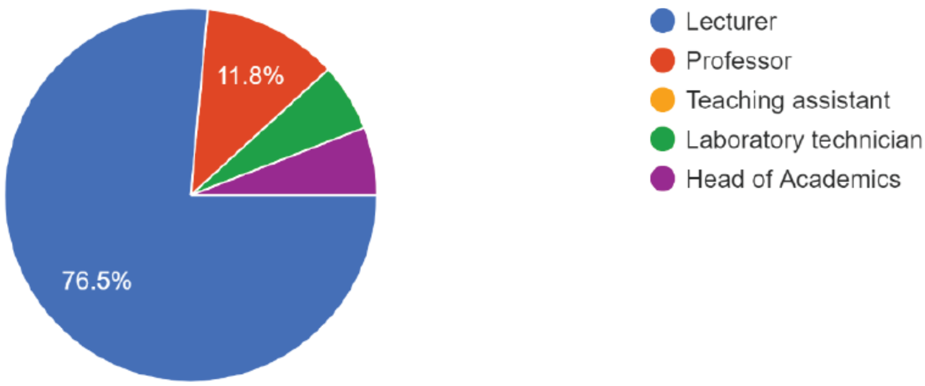


Figure 1: Community of inquiry framework (Garrison and Arbaugh 2007) [14]

3. METHODOLOGY

Denzin and Lincoln (2005) [19] note that the qualitative research approach is useful for providing insights into problems and helps develop ideas. This is our research agenda; we utilize the qualitative research approach, (please see Appendix A: Questionnaire). This is done by collecting data through an online questionnaire from higher education teaching staff, that includes, lecturers, laboratory technicians, heads of academics and professors as shown in Figure 2 below. Of the 17 respondents, about 70% are male and 30% females. These are teaching staff who conducted teaching during the COVID-19 pandemic in developing countries namely Egypt, Zimbabwe, South Africa, Tanzania, Namibia and Botswana.



http://dx.doi.org/10.21622/ACE.2021.01.2.037

Figure 2: Proportion of respondents

The education levels of respondents are 52,9% PhD, 35,3% Masters, 5,9% bachelor and 5,9% Diploma. This shows that the respondents are well educated and fit for our research interests of understanding innovative technologies and practices in times of crisis. In addition, 82,4% of the respondents have more than five years higher education teaching experience.

4. RESULTS

In highlighting the results, first we highlight efforts to promote qualities of learning in the remote learning context. Secondly, we highlight the theoretical contribution and finally explain the practical implications.

4.1. Qualities of learning in the remote learning context

4.1.1 Active learning

Active learning refers to a broad range of teaching strategies which engage students as active participants in their learning during class time with their instructors. Active learning is generally defined as any instructional method that engages students in the learning process. In short, active learning requires students to engage in meaningful learning activities and reflect on these activities. Active learning provides an opportunity for all students in a class to think, engage, apply and synthesize the learning material and come up with solutions using the taught material. It is based on a concept that puts the student at the center of the learning process. The student learns tasks through actively participating. Prior to the outbreak of COVID-19, instructors would organize active learning tasks in class and they would monitor the learners in the execution of these tasks. The 'new normal' has ushered in some drastic changes to this as these tasks are now done outside of the traditional classroom. Lecturers who were interviewed indicated that the 'new normal' and the use of ICT in education promoted active learning. For example, please see respondent quote below:

"Online e-learning approach was used which involved using various electronic devices (e.g. computers, laptops, smartphones, etc.) with internet availability in synchronous or asynchronous environmental conditions. This approach of e-learning provides a platform that makes the process of education more student-centered, creative, and flexible".

4.1.2 Constructive learning

Mulyati et al. (2021, p. 6400) [20] note that "constructivism is the process of building or compiling new knowledge in students' cognitive structures based on experience". In remote learning, this would be promoted by cognitive presence. But there are challenges in remote learning, for instance, amongst the challenges cited was that at times it is difficult to know whether the task was done by the student or not since this is done at home. Activities, such as class discussion, small group discussion, debate, posing questions to the class, think-pair-share activities, short written exercises and polling the class (Bonwell and Eison, 1991) [21] promote constructive learning. It is important to note that the lack of and limited access to learning technologies experienced during a crisis particularly in the developing world is a barrier to constructive learning. About 70% of respondents suggested that students had good access to learning technologies.

4.1.3 Collaborative learning

Collaborative learning involves learners or trainees working in pairs or small groups to discuss concepts or to work out solutions to given problems or tasks. The adoption of the virtual classroom in most developing countries has come with some challenges to the concept of collaborative learning. While it is relatively easy to collaboratively learn in developed countries due to developments in internet infrastructure, in developing countries this concept has faced some challenges to those who have attempted to implement it. In response to COVID-19, institutions of higher learning have come up with strategies meant to enable learning to take place even during the lockdowns. Most institutions prior to COVID-19, conducted lectures physically. This made collaborative learning easy to organize as students were put in groups where they would physically meet and solve issues. In most developing countries where the rate of access to internet infrastructure- and connectivity outside the university campus is very low, collaborative learning online has not been easy. Results from this research indicated that 70 % of lecturers indicated that group work assignments were not submitted on time on the online platform as compared to the physical platform.

4.1.4 Intentional learning

Intentional learning recognizes the efforts of students to actively and willingly achieve cognitive objectives. There were differing levels of support offered by higher education institutions during the COVID-19 pandemic. The respondents noted such measures as training, internet access, mobile gadgets and data bundles. This support creates an appropriate environment for intentional learning. However, a respondent noted the need for "a holistic approach which takes both students and lecturers' needs" for remote learning to achieve intentional learning.

4.1.5 Contextual learning

Contextual learning includes tasks derived from the real world and could involve techniques as case-based or problem-based real-life examples. Learning during a crisis provides unique context that could be taken advantage of in informing the learning process and outcomes. For example, remote learning brought up technology limitations that students could take up as cases and problems meant to be solved as part of the learning. However, a respondent noted that remote learning "was not an easy transition due to resource limitations" but there were efforts "to allow people to adjust to the new normal".

4.1.6 Transfer learning

"Learners are able to transfer learning from the situations and contexts, where learning has taken place and use their knowledge in other situations" ("Author", 2018) [11]. Transfer learning is demonstrated by students during assessment. Assessment during remote learning was a challenge and called for revisions of assessment tasks and techniques to suit the online assessment environment.

4.1.7 Reflective learning

Schön (1983) [22] describes reflective practice as a dialogue between thinking and doing through which the learner becomes more skilled. Institutions of higher learning aim to produce learners who are skilled. In Zimbabwe, for example, the Ministry of Higher and Tertiary Education has adopted a 5.0 education thrust which among other things focuses on innovation, industrialization and research. This places reflective education at the center of such learning. This means that as institutions reacted and continue to react to COVID-19, the ultimate aim is to produce graduates who are skilled. Thus, the adoption of the virtual learning platform and other strategies should support this vision.

4.2 Theoretical contributions

Innovative technologies and practices to the rapid shift to remote learning and

 Volume 1, Issue 2, December 2021 - ISSN 2735-5985
http://dx.doi.org/10.21622/ACE.2021.01.2.037

ensuring qualities of learning are illustrated in Figure 3. We noted that innovative remote learning technologies, such as, learning management systems like Moodle and communication platforms like WhatsApp were used in innovative practices that are student-centered, creative and flexible to create an appropriate leaning environment as defined by the community of inquiry model to promote qualities of learning. There is need to adjust the teaching process to remote learning using for instance community of inquiry principles (Garrison and Anderson, 2003) [23]. This calls for inclusion of social presence, teaching presence and cognitive presence. There are various technological requirements and skills for remote learning, for instance learning management system support and digital pedagogy. These are essential for successful remote learning. In addition, there is increased workload in remote learning.

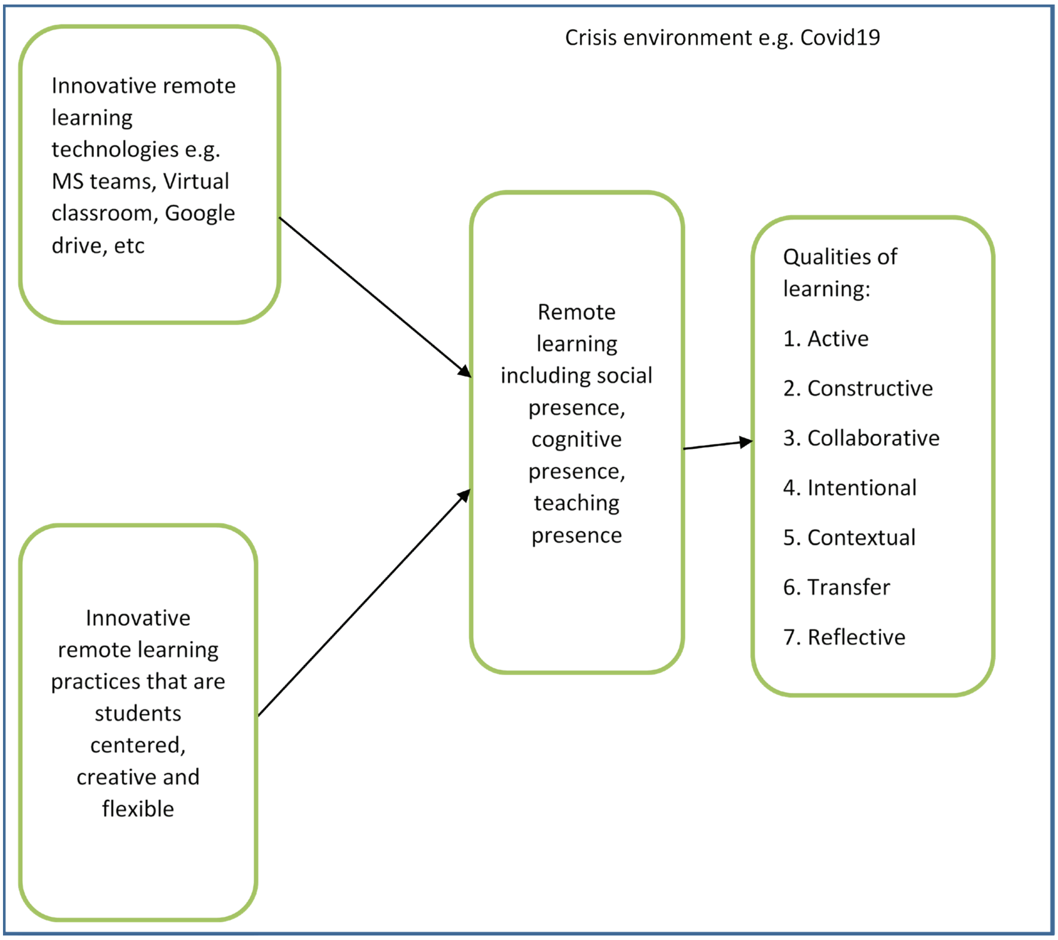


Figure 3: Innovative technologies and practices to the rapid shift to remote learning and ensuring qualities of learning

The perceived levels of qualities of learning during the remote learning during a crisis, specifically COVID-19 by the respondents is illustrated in Figure 4.

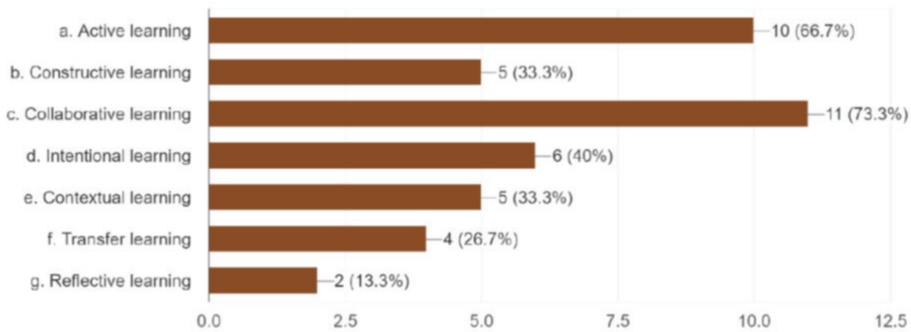


Figure 4: Perceived qualities of learning in the "new normal"

http://dx.doi.org/10.21622/ACE.2021.01.2.037

4.3 Practical implications

This study shows that during times of crisis, innovative technologies and practices are essential to promote learning. In developing countries' context, making use of existing infrastructure, systems and organizations is necessary to promote learning in higher education. COVID-19 has provided opportunities for institutions and individuals to embrace technology and improve service delivery. Even in those institutions that were pessimistic about embracing the use of ICT, the COVID-19 induced lockdowns have changed the way ICT is perceived and utilized.

5. DISCUSSION

Perception of the 'new normal'

A new normal can be described as a state to which a society settles following a crisis, when this differs from the situation that prevailed prior to the start of the crisis. In this case the 'new normal' is a situation in which teaching and learning is not restricted to the brick-and-mortar classroom. In the 'new normal', learning is largely virtual and ICT based. This 'new normal' also means that the time in which teaching and learning happens is flexible as compared to the pre-COVID-19 period in which there were fixed times in which learning was done. This research observed that most of the lecturers interviewed appreciated the new normal that has been ushered in by the outbreak of COVID-19.

Technological changes used in the 'new normal'

The proliferation of mobile devices in institutions of higher learning has radically shaped andragogical and pedagogical aspects of learning. A lot of technological innovations have been introduced in education. Mobile phones, laptops, tablets, radios and T.Vs have been adopted and used in universities and other institutions of learning in most developing countries. Mobile learning (m-learning) has been increasingly incorporated into formal classroom delivery in various universities (Schuck, Kearney, & Burden, 2017) [24].

While mobile phones are known to be very popular among university students for increasing their social inclusion and connectedness as well as providing a sense of security as they can contact others in times of distress or emergency (Balakrishnan & Raj, 2012) [25], the outbreak of COVID-19 has resulted in the use of the mobile phones as one of the major teaching and learning devices. COVID-19 has promoted the increased adoption of online learning in universities and most private schools in developing countries. Universities have encouraged their students to bring their own learning devices to class.

The Midlands State University in Zimbabwe has adopted a Buy Your Own Device (BYOD) policy to facilitate m-learning (Pondiwa and Phiri, 2019) [26]. While students have always been bringing their own ICT gadgets to university before the outbreak of COVID-19, the outbreak of COVID-19 and the subsequent lockdowns that followed increased the need to for students to have their own devices.

Universities responded to the COVID-19 outbreak in several ways. The better resourced ones were quick to adjust by closing and quickly coming up with alternatives of virtual teaching and learning. Some gave their students pre-recorded learning materials while others offered online learning with learners getting lecture material in real time. Other universities provided their lecturers with teaching materials such as internet data bundles and devices for lecture deliveries.

http://dx.doi.org/10.21622/ACE.2021.01.2.037

6. CONCLUSION

This research shows innovative technologies and practices to promote learning in higher education during a crisis e.g. COVID-19 pandemic. Existing technology infrastructure e.g. mobile phone and applications such as WhatsApp were used for learning purposes in higher education. Institutions have adapted and continue transforming their way of doing business. There has been an increase in the number of organizations that offer online services. Educators and learners have also transformed educational systems in order to cope with requirements of learning during a crisis, such as COVID19 pandemic.

REFERENCES

[1] Jacob Martinez, "Take this pandemic moment to improve Education", Internet:https://edsource.org/2020/take-this-pandemic-moment-to-improve, June, 2020.

[2] Rapanta, C., Botturi, L., Goodyear, P. et al., " Online University Teaching During and After the Covid-19 Crisis: Refocusing Teacher Presence and Learning Activity." Postdigital Science and Education, vol. 2, pp.923–945, July 2020.

[3] UNESCO, "COVID-19 educational disruption and response." Internet: https://en.unesco.org/themes/education-emergencies/ coronavirus-school-closures, March, 2020

[4] Shivangi Dhawan, "Online Learning: A Panacea in the Time of COVID-19 Crisis", Journal of Educational Technology Systems 2020, Vol. 49(1) 5–22.

[5] Mishra, L., Gupta, T., & Shree, A.," Online teaching-learning in higher education during lockdown period of COVID-19 pandemic." International Journal of Educational Research Open, Vol.1, 100012, 2020.

[6] Wu, Z, "How a top Chinese university is responding to coronavirus. World Economic." Internet: https://www.weforum.org/agenda/2020/03/coronavirus-china-the-challenges-of-onlinelearning-for-universities/, . 16,March,2020, [accessed 13 July 2020]

[7] Shireen Motala,Kirti Menon, "In search of the 'new normal': Reflections on teaching and learning during Covid-19 in a South African university." Southern African Review of Education with Education with Production, Vol. 26, No. 1, pp.185 – 195, July 2020

[8] Baran, Evrim," The transformation of online teaching practice: Tracing successful online teaching in higher education.", Doctoral thesis, Iowa State University, United States, 2011.

[9] Victor C. X. Wang1Geraldine Torrisi-Steele, "Online Teaching, Change, and Critical Theory. New Horizons in Adult Education & Human Resource Development, Wiley Periodicals, Inc., Vol. 27, Issue3, pp. 18-26, 2015

[10] Victor C. X. Wang1Geraldine Torrisi-Steele, "Online Teaching, Change, and Critical Theory. New Horizons in Adult Education & Human Resource Development, Wiley Periodicals, Inc., Vol. 27, Issue3, pp. 18-26, 2015

[11] Mavengere, N., Ruohonen, M.,"Context and user needs in virtual learning in pursuit of qualities of learning," Education and Information Technologies, Vol.3, Issue 4, pp.1607-1620, July 2018

[12] Hartikainen S, Rintala H, Pylväs L, Nokelainen P. "The Concept of Active Learning and the Measurement of Learning Outcomes: A Review of Research in Engineering Higher Education." Education Sciences, Vol. 9, issue 4, pp. 279, 2019. https:// doi.org/10.3390/educsci9040276

[13] Ruokamo, H., & Pohjolainen, S.," Pedagogical Principles for Evaluation of Hypermedia-Based Learning Environments in Mathematics." JUCS–Journal of

http://dx.doi.org/10.21622/ACE.2021.01.2.037

Universal Computer Science, Vol.4, Issue 3, pp. 292–307, 1998.

[14] Garrison, D. R., & Arbaugh, J. B.," Researching the community of inquiry framework: Review, issues, and future directions." The Internet and Higher Education, Vol.10, Issue 3,pp. 157–172, 2007.

[15] Lowenthal, D. A., & Lowenthal, P. R.," A mixed methods examination of instructor social presence in accelerated online courses." Paper presented at the annual meeting of the American Educational Research Association, Denver, CO.2010. doi: 10.4018/978-1-4666-9582-5.ch006.

[16] Garrison, D. R., Anderson, T., & Archer, W.," Critical inquiry in a text based environment: Computer conferencing in higher education." The Internet and Higher Education, Vol. 2, pp. 87-105, 2020.

[17] Anderson, T., Rourke, L., Garrison, D. R., & Archer, W. "Assessing teaching presence in a computer conference context." Journal of Asynchronous Learning Networks,Vol. 5, Issue 2, 2001.

[18] Fiock H.S. "Designing a Community of Inquiry in Online Courses," International Review of Research in Open and Distributed Learning, Volume 21, pp 134-152, 2020.

[19] Denzin, N.K. and Lincoln, "Introduction: The Discipline and Practice of Qualitative Research." Handbook of Qualitative Research, 3rd Edition, Sage, Thousand Oaks, pp.1-32, 2005.

[20] T.S. Mulyati and E. Yetti, Analysis Of Constructive Learning Models In Forming Religious Characters Of Early Childhood. Psychology and Education Journal, 58(2), pp.6399-6403, 2021.

[21] Bonwell, C.C. and Eison, J.A., " Active Learning: Creating Excitement in the Classroom", ASHE-ERIC Higher Education Reports. ERIC Clearinghouse on Higher Education, The George Washington University, One Dupont Circle, Suite 630, Washington, DC 20036-1183, 1991.

[22] Donald A. Schon and Vincent DeSanctis , The Reflective Practitioner: How Professionals Think in Action." The Journal of Continuing Higher Education, 34(3), pp. 29–30, 1983.

[23] D.R. Garrison, T. Anderson, and W. Archer, A theory of critical inquiry in online distance education. Handbook of distance education, 1(4), pp.113-127, 2003.

[24] S. Schuck, M. Kearney, K. and Burden, Exploring mobile learning in the third space. Technology, Pedagogy and Education, 26(2), pp.121-137, 2017.

[25] V. Balakrishnan and R. G. Raj, Exploring the relationship between urbanized Malaysian youth and their mobile phones: A quantitative approach. Telematics and Informatics, 29(3), pp.263-272, 2012.

[26] S. Pondiwa, and M. Phiri, M., Challenges and Opportunities of Managing Social Media Generated Records in Institutions of Learning: A Case of the Midlands State University, Zimbabwe. In International Conference on Sustainable ICT, Education, and Learning (pp. 145-156). Springer, Cham, 2019.

APPENDIX A - QUESTIONNAIRE / INTERVIEW QUESTIONS

Demographic questions

1. What is the name of your university/college?
2. What is the name of your department?
3. How long have you worked here?
4. What is your current role/position in the organisation?
5. In which country is your university / college geographically located?
6. What is your gender? – Male or Female
7. What is your level of education? – Certificate, Bachelor, Masters, PhD

Volume 1, Issue 2, December 2021 - ISSN 2735-5985
http://dx.doi.org/10.21622/ACE.2021.01.2.037

Research questions

1. What are the practices used for learning in higher education during the COVID19 pandemic in developing countries?
2. What are the technologies (this could include mobile based, TV, radio, etc.) used for learning in higher education during the COVID19 pandemic in developing countries?
3. How accessible were these technologies?
4. How was the transition done to continue learning in higher education during the COVID19 pandemic in developing countries?
a. Why was the transition done this way?
5. What were the facilitating conditions offered by the university?
6. What are the lessons i.e. adaptable approaches that could be used to transition to remote learning during a crisis?
7. What is your perceived "New Normal" in Higher Education based on the Innovative teaching and learning technologies and practices experience during COVID19
8. How does the "new normal" promote
    a. Active learning
    b. Constructive learning
    c. Collaborative learning
    d. Intentional learning
    e. Contextual learning
    f. Transfer learning
    g. Reflective learning

http://dx.doi.org/10.21622/ACE.2021.01.2.047

A Drop-in Replacement for LR(1) Table-Driven Parsing

Michael Oudshoorn

Founding Dean, High Point University, School of Engineering

One University Parkway, High Point, NC 27268, USA

Email: {moudshoo@highpoint.edu}

Received on, 27 April 2021 - Accepted on, 10 July 2021 - Published on, 15 November 2021

ABSTRACT

This paper presents a construction method for a deterministic one-symbol look-ahead LR parser which allows non-terminals in the parser look-ahead. This effectively relaxes the requirement of parsing the reverse of the right-most derivation of a string/sentence. This is achieved by replacing the deterministic push down automata of LR parsing by a two-stack automata. The class of grammars accepted by the two-stack parser properly contains the LR(k) grammars. Since the modification to the table-driven LR parsing process is relatively minor and mostly impacts the creation of the *goto* and *action* tables, a parser modified to adopt the two-stack process should be comparable in size and performance to LR parsers.

Keywords: LR Parsing, Parsing Algorithm.

1. Introduction

The LR parser [1], characterized as a one-symbol look-ahead parser [2], is based on a deterministic push-down automata (DPDA). A strength of this technique is that the same parser engine can be used with parse tables generated by any of the increasingly general LR(0), SLR(1), LALR(1) and LR(1) algorithms. Parsing is limited to LR(1) by the parser mechanism. Although LR parsers accept the deterministic context-free languages (DCFLs) [3, 4] some context-free grammars (CFGs) for DCFLs require significant transformation before they can be used in LR parser construction. Seite [5] observes that such transformations are tedious and may reduce readability of the grammar, or generate a grammar that is not exactly equivalent to the original. However, a parser generator accepting a wider class of grammars evades many of these disadvantages by simply avoiding the need to transform the grammar.

This paper discusses a one-symbol look-ahead parser employing two stacks as a practical replacement for the LR parsing engine. The engine is based on a deterministic two-stack machine [6] and other two-stack (bottom-up) parsers [3, 7, 8]. A parser construction method for the two-stack machine which produces efficient, deterministic parsers for a broad class of grammars properly including LR(k) grammars is presented. For non- LR(k) grammars, the parser can defer parsing decisions while an arbitrary sequence of input symbols is processed.

http://dx.doi.org/10.21622/ACE.2021.01.2.047

Unbounded forward parsing activity is converted into non-terminals and is not repeated. This allows the parsing of a class of non- LR(k) grammars using a single (terminal or non-terminal) symbol look-ahead.

Without formal proof, it is asserted that the generated parsers preserve the desirable properties of LR parsing, namely acceptance of a broad class of practical grammars, error detection at the first invalid symbol (hence guaranteed termination) [8] and $O(n)$ parsing efficiency (both time and space).

Section 2 introduces the terminology and notation used throughout the paper. Section 3 introduces an example that highlights some of the deficiencies of LR parser construction. Section 4 examines the two-stack parsing process and Section 5 presents the basic parser construction. Section 6 extends the parser generator for increased generality Results are presented in Section 7 and conclusions are offered in Section 8.

2. Terminology

Formally, a context-free grammar is a tuple G = <N, T, P, S> where N and T are disjoint sets of, respectively, non-terminal and terminal symbols (hence $N \cap T = \varnothing$), P is a finite set of productions, and $S \in N$ is the goal symbol. $V = N \cup T$ is defined as the vocabulary of the grammar. A production is a pair, written $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in V^*$. The following conventions are used in this paper:

$$
\begin{array}{rl}
S,\ A,\ B,\ C\ \ldots & \in N \\
\ldots\ X,\ Y,\ Z & \in V \\
\alpha,\ \beta,\ \gamma\ \ldots & \in V^* \\
a,\ b,\ c,\ \ldots & \in T \\
\ldots\ ,\ x,\ y,\ z & \in T^*
\end{array}
$$

The notation, ε, represents an empty sequence and the length of a sequence α is $|\alpha|$, hence $|\varepsilon| = 0$.

The relation $\Rightarrow$, pronounced "directly produces", is defined on V^* such that $\alpha A \gamma \Rightarrow \alpha\beta\gamma$ where $A \rightarrow \beta \in P$. The relation closures $\Rightarrow^*$ and $\Rightarrow^+$, both pronounced "produces", are used to define the sentential forms, $\{\alpha \mid S \Rightarrow^* \alpha\}$, of the grammar G including the (probably infinite) language generated, $L(G) = \{x \mid S \Rightarrow^+ x\}$.

3. An exercise in parsing

Backus-Naur form *(BNF)* is a notation [9] describing the productions of a grammar. BNF is a small language that provides parsing problems for conventional parsing technologies.

The grammar G_1 (Figure 1) is a natural syntax definition of BNF notation without options in the right-hand side of productions. The language $L(G_1)$ = BNF is used as the primary language example in this paper. In grammar G_1, left recursion is used to describe a non-empty sequence S of productions. Each production P is a sequence $n ::= R$ where R is a possibly empty sequence of terminal t and non-terminal n symbols. A typical parsing system will distinguish terminal and non-terminal symbols in a separate lexical analysis phase.

An attempt to build a LALR(1) or LR(1) parser from this grammar fails, reporting a shift/reduce conflict. For some grammars, resolving such a conflict in favor of the shift action effectively disambiguates the grammar [10]. This strategy fails to produce a BNF parser based on G_1 because G_1 is not (suitably) ambiguous. BNF, though not strictly a programming language, is a simple and relevant language which shares characteristics with many computer languages. The failure of LR(1) parser generation for G_1 is disappointing given the expectations built up in the literature:

- "*Most unambiguous grammars for programming languages are SLR(1) – hence LALR(1) and LR(1) – in the form in which they are originally written.*" [11]
- "*This class of grammars [LR(1)] has great importance for compiler design, since they are broad enough to include the syntax of almost all programming languages, yet restrictive enough to have efficient parsers that are essentially DPDA's.*" [6]
- "*Theoretically, LR(k) parsers are of interest in that they are the most powerful class of deterministic bottom-up parsers using at most k look-ahead symbols. … if a grammar, G, can be parsed by any deterministic bottom-up parser using k look-ahead symbols, a LR(k) parser can be built for G.*" [12]

$$
\begin{aligned}
&(1)\ S \rightarrow S\,P\\
&(2)\ S \rightarrow P\\
&(3)\ P \rightarrow n ::= R\\
&(4)\ R \rightarrow \varepsilon\\
&(5)\ R \rightarrow R\,n\\
&(6)\ R \rightarrow R\,t
\end{aligned}
$$

Figure 1. Numbered productions of grammar G1

More precisely, it is well known that all, and only, the deterministic CFL's have LR(1) grammars [6] While parsers can be developed for a wide class of languages, but they are not always based on a grammar that relates naturally to the semantics. Knuth [4] made it clear from the start that:

"… LR(k) is a property of the grammar, not of the language alone. The distinction between grammar and language is extremely important when semantics is being considered as well as syntax."

BNF is an example of such a language. BNF is a regular language (RL), so $L(G_1) \in RL \subset DCFL$ and can be described by the equivalent regular expressions

http://dx.doi.org/10.21622/ACE.2021.01.2.047

in Figure 2. From the latter regular expression, $G_2 \in LR(1)$ where $L(G_1) \equiv L(G_2)$ (see Figure 3) can be constructed.

$$\begin{aligned} (n ::= (n \mid t)^*) + \quad &\equiv n ::= (n \mid t)^* \, (n ::= (n \mid t)^*)^* \\ &\equiv n ::= ((n \mid t)^* \, n ::=)^* \, (n \mid t)^* \end{aligned}$$

Figure 2. Regular expressions for BNF.

$$\begin{aligned} &S \rightarrow P\ R \\ &P \rightarrow n ::= \mid P\ R\ n ::= \\ &R \rightarrow \varepsilon \mid R\ n \mid R\ t \end{aligned}$$

Figure 3. Grammar $G_2 \in LR(1)$ and $L(G_2) \equiv$ BNF.

The link between the syntax and the semantics of G_2 is less obvious than with G_1: namely, each R is associated with the last $n ::=$ in the preceding P. Grammar G_2 is arguably a less natural expression, albeit an equivalent, of the grammar for BNF.

A language designer may also alter the language as well as the grammar to suit a class of parser generator, e.g. by adding terminators and/or separators [13, 14] as in grammars G_{3a}, G_{3b} and G_{3c} (Figure 4). This syntactic notation is reminiscent of statement syntax in programming languages such as Algol-[9], C [15], Pascal [16], Modula-2 [17], Modula-3 [18], Ada [19, 20], etc. It is also argued that such notation aids error recovery and is simply good language design practice. Other languages, e.g. BCPL [21] and Haskell [22, 23], adopt this solution but try to use layout to mask the consequences.

$S \rightarrow S\ ;\ P \mid P$	$S \rightarrow S\ P \mid P$	$S \rightarrow S'\,semi_{opt}$
$P \rightarrow n ::= R$	$P \rightarrow n ::= R\ ;$	$S' \rightarrow S'\ ;\ P \mid P$
$R \rightarrow \varepsilon \mid R\ n \mid R\ t$	$R \rightarrow \varepsilon \mid R\ n \mid R\ t$	$P \rightarrow n ::= R$
		$R \rightarrow \varepsilon \mid R\ n \mid R\ t$
		$semi_{opt} \rightarrow \varepsilon \mid ;$
G_{3a}: separator	G_{3b}: terminator	G_{3c}: separator/terminator

Figure 4. Grammars for BNF with terminators and/or separators.

Since G_1 cannot be parsed using LR(1) parsers, the more general LR(k) techniques can be considered. This immediately meets with success since $G_1 \in LR(2)$. An experienced language design could manipulate the grammar to produce grammar $G_{4a} \in LR(1)$ (Figure 5) by treating $n ::=$ as the single lexical item, *lhs*. Alternatively, using the forward context capabilities of a lexical analysis tool such as [24, 25], the symbol n followed by $::=$ can be distinguished from n followed by some other symbol, before it reaches the parser.

http://dx.doi.org/10.21622/ACE.2021.01.2.047

A general parser generator could allow the separation of a language into lexical and syntactic elements as a matter of style or taste rather than necessity. A comparison of G_{4a} with G_2 suggests that the lexical/syntactic division can be important in the preservation of semantic structure during grammar development. Grammar G_{4b} (Figure 5) for Algol-60 BNF BNF [9] is based on another lexical/syntactic separation and is as "natural" as G_1. Unfortunately, G_{4b} is not LR(2) nor even LR(k); from such a starting point, a lexical solution to the parsing problem may be more difficult to discover.

$$S \to SP \mid P \qquad P \to lhs\ R \qquad R \to \varepsilon \mid R\ n \mid R\ t$$

$$S \to SP \mid P \qquad P \to N ::= \qquad P \to\ P\ N \mid Pt \qquad N \to < C > \qquad C \to c \mid C\ c$$

Grammar $G_{4a} \in$ LR(1). Grammar $G_{4b} \in$ LR(k).

Figure 5. Alternate syntax/lexical separations.

The essence of this difficulty is that LR grammars have poor closure behavior under homomorphism [26,6] and substitution with regular expressions. Syntactic difficulties detected in parser generation can be treated as semantic issues or transferred to the lexical analyzer. The limited capabilities of most lexical analyzers and the increased costs (space and/or time) diminish the applicability of the latter strategy. The two-stack parser generator presented in this paper is less sensitive to this division of responsibility between the lexical analysis and parsing modules.

Grammars G_{5a} and G_{5b} (Figure 6) represents grammars that can be parsed with a two-stack mechanism where $L(G_{5a}) \equiv L(G_{5b}) \equiv$ BNF. LR(1) parser generators, working exclusively with terminal-symbol look-ahead fail for G_{5a} and G_{5b}. With G_{5b}, the sequence $R\ n$ can be derived from either R or $R\ L$ resulting in a non-deterministic parser. Further, the LR parser cannot reverse a derivation step, $P\ L \Rightarrow L ::= R\ L$, as the trailing L would be overwritten in the parsing engine. This illustrates the differences between LR parsing and the two-stack technique. By using non-terminal look-ahead, the two-stack parser can reverse both G_{5b} derivation sequences:

$$S \to P\ S \mid P \qquad P \to n ::= R \qquad R \to \varepsilon \mid R\ n \mid R\ t$$

Grammar G_{5a}

$$S \to S\ P \mid P \qquad P \to L ::= R \qquad R \to \varepsilon \mid R\ n \mid R\ t \qquad L \to n$$

Grammar G_{5b}

Figure 6. BNF grammars for two-stack parsing.

1. $P\ L ::= \ldots \Rightarrow L ::= R\ L ::= \ldots \Rightarrow L ::= R\ n ::= \ldots$
2. $P\ \ldots \Rightarrow\ ::= R\ \ldots \Rightarrow L ::= R\ n\ \ldots \Rightarrow\ L := R\ n\ n\ \ldots$

http://dx.doi.org/10.21622/ACE.2021.01.2.047

The technique described in Section 5 develops a parser using two-stack automata via a construction generalized from LR parser generation. The parser generator presented fails for G_1, but succeeds for G_{5a}, G_{5b} and G_6 (Figure 7). Each of these grammars might be described as natural relative to G_2. Grammar G_6, like G_{4b}, is a BNF grammar without empty productions, and is the basis for the example two-stack BNF parser developed in Sections 4 and 5.

$$
\begin{array}{ll}
(1) & S \rightarrow S\ P \\
(2) & S \rightarrow P \\
(3) & P \rightarrow n\ ::= \\
(4) & P \rightarrow P\ n \\
(5) & P \rightarrow P\ t
\end{array}
$$

Figure 7. Grammar G_6 with numbered productions.

The parser generator described below is more general than the LR(1) algorithm and more practical than for the deterministic regular parsable (DRP) grammars [8]. The two-stack parser is simpler than parsers for other developments from LR(1), e.g. LR(k) and LR(k, t) [4], LRR [26, 5], LAR(m) [28] and DRP.. Figure 8 shows the capabilities of the parsers for the BNF-describing grammars given above.

	G_1	G_2	G_{4a}	G_{4b}	G_{5a}	G_{5b}	G_6
LR(1)		√	√				
LR(2)/LR(k)	√	√	√		√	√	√
Two-stack		√	√	√	√	√	√

Figure 8. Grammar class membership.

4. A Two-Stack Parser

The two-stack parser has a left-stack containing integer-subscripted symbols and a right-stack containing symbols. The set of symbols, V, comes from a grammar and the integers from the set, K, of node identifiers from the parser construction. The stacks are delimited by a special $\perp$ symbol and are illustrated below with stack-tops toward the center.

The parser has four actions:

- *Shift i* where $i \in K$, pops a symbol, X, from the right-stack and pushes the subscripted symbol, X_i, into the left-stack. Pictorially:

left stack	right stack		left stack	right stack
$\perp_0 \ldots Y_j$	$X\ \alpha \perp$	*shift i* $\rightarrow$	$\perp_0 \ldots Y_j\ X_i$	$\alpha \perp$

- *Reduce* $\alpha \rightarrow \beta$ pops β', an annotated version of β, from the left-stack and pushes into the right-stack. The notation *Reduce* $n + m$ is used for *Reduce* $A\mu \rightarrow \alpha\mu$ where $P_n = A \rightarrow \alpha$ and $m = |\mu|$, or simply *Reduce* n when $\mu = \varepsilon$ (m = 0).

http://dx.doi.org/10.21622/ACE.2021.01.2.047

left stack	right stack		left stack	right stack
$\perp_0 \ldots X_I\ \alpha'\ \mu'$	$\gamma \perp$	*reduce* $n + m \rightarrow$	$\perp_0 \ldots X$	$A\ \mu\ \gamma \perp$

- *Accept* causes successful termination.
- *Error* causes failure termination.

The parser uses the subscript and symbol at the top of the stacks to access the parse table, a function $PT: K \times V \rightarrow action$, to determine the action to be taken at each step. This table is represented with *Error* entries left blank, e.g. Figure 9 shows a parser table derived from grammar G_6.

Node	$\perp$	n	t	$::=$	S	P
0		S3			S2	S1
1	R2	S4	S5			R2
2	Acc	S3				S6
3				S8		
4	R4	R4	R4	S7		R4
5	R5	R5	R5			R5
6	R1	S4	S5			R1
7	R3	R3	R3			R1
8	R3	R3	R3			

Figure 9. Parse table for grammar G_6.

Initially, the left-stack holds the $\perp_0$ symbol and the right-stack holds the input string followed by the $\perp$ marker. Valid input is accepted in a state where the left-stack contains the goal symbol and the right-stack is empty.

left stack	right stack	
$\perp_0$	*input* $\perp$	*action* $\rightarrow \ldots$

	left stack	right stack	
$\rightarrow$ *action* $\rightarrow$	$\perp_0\ S_i$	$\perp$	*Accept*

With invalid input, the parser reaches an *Error* action. The complete sequence of parser actions for a short string from $L(G_6)$ is shown in Figure 10.

Provided the left-stack is correctly conditioned for each reduce (as in LR parsing), partial correctness of the parser can be shown by reverse induction. At termination, the stacks contain a $\perp$ delimited sentential form. Shift preserves a sentential form in the stacks, hence it can be ignored. Reduce is effectively the inverse of $\Rightarrow$; a *reduce* action resulting in a sentential form necessarily starts from a sentential form. A successful parse, a sequence of *shift* and *reduce* actions terminating with *accept*, must commence with a sentential form in the stacks. Responsibility for ensuring that the parser moves forward from an accessible parser state rests with the parser construction method.

left stack	*right stack*	*action*
$\perp_0$	$n ::= n\ t\ n ::= n \perp$	*Shift 3*
$\perp_0\ n_3$	$::= n\ t\ n ::= n \perp$	*Shift 8*
$\perp_0\ n_3\ ::=_7$	$n\ t\ n ::= n \perp$	*Reduce 3*
$\perp_0$	$P\ n\ t\ n ::= n \perp$	*Shift 1*
$\perp_0\ P_1$	$n\ t\ n ::= n \perp$	*Shift 4*
$\perp_0\ P_1\ n_4$	$t\ n ::= n \perp$	*Reduce 4*
$\perp_0$	$P\ t\ n ::= n \perp$	*Shift 1*
$\perp_0\ P_1$	$t\ n ::= n \perp$	*Shift 5*
$\perp_0\ P_1\ t_5$	$n ::= n \perp$	*Reduce 5*
$\perp_0$	$P\ n ::= n \perp$	*Shift 1*
$\perp_0\ P_1$	$n ::= n \perp$	*Shift 4*
$\perp_0\ P_1\ n_4$	$::= n \perp$	*Shift 7*
$\perp_0\ P_1\ n_4\ ::=_7$	$n \perp$	*Reduce 3*
$\perp_0\ P_1$	$P\ n \perp$	*Reduce 2*
$\perp_0$	$S\ P\ n \perp$	*Shift 2*
$\perp_0\ S_2$	$P\ n\perp$	*Shift 6*
$\perp_0\ S_2\ P_6$	$n \perp$	*Shift 4*
$\perp_0\ S_2\ P_6\ n_4$	$\perp$	*Reduce 3*
$\perp_0\ S_2$	$P \perp$	*Shift 6*
$\perp_0\ S_2\ P_6$	$\perp$	*Reduce 1*
$\perp_0$	$S \perp$	*Shift 2*
$\perp_0\ S_2$	$\perp$	*Accept*

Figure 10. Parsing $n ::= n\ t\ n ::= n$ with grammar G_6.

5. Parse table construction

A parser is constructed from a grammar augmented with the parser symbols $\perp \in T$ and $S' \in N$ used in a new production $P_0 = S' \rightarrow \perp S \in P$. The following relations based on the grammar are defined:

$$FRONT(A\alpha) = FRONT(A) = \{X \mid A \Rightarrow X\beta\}$$
$$FRONT_0(X\alpha) = \{X\}$$
$$FRONT(\varepsilon) = \varnothing$$
$$FWD^+(\alpha) = \cup\{X \mid \alpha \Rightarrow^+ X\beta\}$$
$$FWD^*(\alpha) = \cup\{X \mid \alpha \Rightarrow^* X\beta\}$$
$$FIRST_k(\alpha) = \begin{cases} \varepsilon, k = 0 \\ \alpha, \alpha = \perp \vee \alpha = \varepsilon \\ x, \alpha \Rightarrow^* \alpha\beta \wedge x \in \alpha FIRST_{k-1}(\beta) \end{cases}$$

An item is a tuple <n, μ, β, γ>, alternatively represented as $A\mu \rightarrow \alpha \bullet \beta; \gamma$, where $P_n = A \rightarrow \delta \in P$, $\delta\mu = \alpha\beta$, and $\gamma \in V^*$ is an element of a (finite) set of contexts. Informally, an item reads: having found α, look for β and (an initial segment of) context γ before reducing $\alpha\beta$ to $A\mu$. An initial item has the form $A\mu \rightarrow \bullet\ \delta\mu; \gamma$ or $<n, \mu, \delta, \gamma>$ where $P_n = A \rightarrow \delta$. A final item has the form $A\mu \rightarrow \delta\mu \bullet; \gamma$ or $<n, \mu, \varepsilon, \gamma>$. The construction described in this section maintains $\mu = \varepsilon$; the construction in Section 6 produces non-empty values of μ. Items with differing contexts may be displayed as a group, i.e:

$$A\mu \rightarrow \alpha \bullet \beta; \gamma, \ldots, A\mu \rightarrow \alpha \bullet \beta; \delta \equiv A\mu \rightarrow \alpha \bullet \beta; \gamma, \ldots, \delta$$

http://dx.doi.org/10.21622/ACE.2021.01.2.047

Initially, a directed graph (digraph) is constructed where nodes are sets of items and edges are labelled with symbols. The nodes in the digraph are uniquely numbered. The initial node is the singleton item set, $I_0 = \{S' \rightarrow \perp \bullet S; \perp\} \equiv \{<0, \varepsilon, S, \perp>\}$. The digraph for G6 is shown in Figure 11.

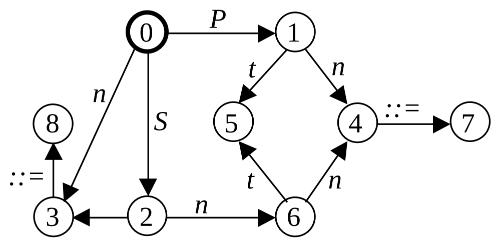


Figure 11. Digraph for grammar G_6.

The parser table develops from the digraph. Each edge in the graph represents a shift action in the table. The reduce and accept entries in the table are based on the final items at a node. The node items for grammar G6 are shown in Figure 12.

Node	Node items
0	$S' \rightarrow \perp \bullet S; \perp$
1	$S \rightarrow P \bullet; \perp, P \perp, P\ n$ $P \rightarrow P \bullet n; \perp, P \perp, P\ n, n, t$ $P \rightarrow P \bullet t; \perp, P \perp, P\ n, n, t$
2	$S' \rightarrow \perp S \bullet; \perp$ $S \rightarrow S \bullet P; \perp, P \perp, P\ n$
3	$P \rightarrow n \bullet ::=; \perp, P \perp, P\ n, n, t$
4	$P \rightarrow n \bullet ::=; \perp, n, t$ $P \rightarrow P\ n \bullet; \perp, P \perp, P\ n, n, t$
5	$P \rightarrow P\ t \bullet; \perp, P \perp, P\ n, n, t$
6	$S \rightarrow S\ P \bullet; \perp, P \perp, P\ n$ $P \rightarrow P \bullet n; \perp, P \perp, P\ n, n, t$ $P \rightarrow P \bullet t; \perp, P \perp, P\ n, n, t$
7	$P \rightarrow n ::= \bullet; \perp, P \perp, P\ n, n, t$
8	$P \rightarrow n ::= \bullet; \perp, n, t$

Figure 12. Item sets of nodes from grammar G_6.

5.1 Digraph Construction

Each node is uniquely numbered and represents a set of items similar to basis items in LR parsing literature. The digraph is constructed by starting from node I_0 and constructing edges and nodes until the reachable nodes have been identified.

http://dx.doi.org/10.21622/ACE.2021.01.2.047

A node I_n has a (possibly empty) set of shift symbols, $SS(I_n)$. Two groups of symbols contribute to the shift symbols. Basic shift symbols, $BS(I_n)$, are derived from non-final items and are the symbols that must be shifted by the parser to progress towards a reduce action. Reduce conflict symbols, $RC(I_n)$, are symbols which lead to multiple reduce entries, i.e. a reduce/reduce conflict, in an LR parser generation. Two-stack parser construction treats these symbols as shift symbols.

In two-stack parsing, the symbols in $RC(I_n)$ are shifted with the expectation that the conflict will be resolved once a non-terminal is recognized. $RC(I_n)$ is the set of symbols which appear in more than one reduce symbol set for an item group, $RS(I_n, n, \mu, \alpha)$. Consider a node containing two final items, $<i, \mu, \varepsilon, \alpha>$, $<i, \mu, \varepsilon, \alpha>$, $<j, \eta, \varepsilon, \beta> \in I_k$, where $\alpha \Rightarrow^* X\,\gamma \Rightarrow Z \ldots$ and $\beta \Rightarrow^* Y\,\delta \Rightarrow Z \ldots$. The parser generator avoids potential parse table conflicts:

$$\{reduce\ i, reduce\ j\} \subseteq PT(I_k, W)$$
$$\text{where } W \in FWD^*(Z) \subset RS(I_k, i, \mu, \varepsilon) \cap RS(I_k, j, \eta, \varepsilon)\ )$$
$$\text{hence } FWD^*(Z) \subseteq RC(I_k) \subseteq SS(I_k)$$

As a result, items $X \to \bullet\, Z \ldots; \gamma$ and $Y \to \bullet\, Z \ldots; \delta$ and their derivatives are included in the node closure. The result is a parser which tries to recognize either X or Y starting with Z. Having reduced X or Y onto the top of the right-stack, the parser is able to decide between reduce i and reduce j.

The same strategy deals with conflicts between shift and reduce actions. The presence of empty productions, productions of the form $A \to \varepsilon$, in the grammar requires that non-final items be included in the definition of RC.

$$SS(I_n) = BS(I_n) \cup RC(I_n) - \{\perp\}$$
$$BS(I_n) = \cup\{FRONT^*(\alpha) \mid <i, \mu, \alpha, \beta> \ \in I_i\}$$
$$RC(I_n) = \cup\{RS(I_n, i, \mu, \alpha) \cap RS(I_n, j, \eta, \beta) \mid <i, \mu, \alpha> \neq <j, \eta, \beta>$$
$$RS(I_i, n, \mu, \alpha) = \cup\{FWD^*(\alpha\ \beta) - FRONT^*(\alpha) \mid <n, \mu, \alpha, \beta> \in I_i\}$$

Every digraph node has a labelled edge leaving it for each shift symbol associated with that node, hence:

$$Edges(I_n) = \{<I_n, X, Next(I_n, X)> \mid X \in SS(I_n)\}$$
$$Next(I_i, X) = \{<n, \mu, \alpha, \beta> \ \mid <n, \mu, X\,\alpha, \beta> \ \in \ Close(I_i)\}$$

Each node has a (possibly empty) set of closure items. These are computed by recursively extending the set of items that may be applicable in the (local) parsing position represented by the node. A complete set of closure items is identified, prior to the relevant items being selected, as this simplifies development of item contexts.

http://dx.doi.org/10.21622/ACE.2021.01.2.047

$$Close(I_n) = I_n \cup Relevant(SS(I_n), Derive^{+}(I_n))$$

$$Derive(I_i) = \cup \begin{cases} CI(A, \delta), <n, \mu, \alpha A\beta, \gamma> \in I_I \wedge \alpha \Rightarrow^* \varepsilon \wedge \delta \in Context(\beta, \gamma) \\ CI(A, \gamma), \delta), <n, \mu, \alpha A\beta, \beta A\gamma> \in I_I \wedge \alpha\, \beta \Rightarrow^* \varepsilon \end{cases}$$

$$CI(A, \beta) = \{<n, \varepsilon, \alpha, > \mid P_n = A \rightarrow \alpha\}$$

$$Context(\alpha, \beta) = \begin{cases} \beta, \alpha = \varepsilon \\ \gamma\, \alpha, \gamma\, \alpha\, \delta = \alpha\, b \wedge \gamma \in N^* \wedge b \in FIRST_1(\beta) \end{cases}$$

$$Relevant(R, Q) = \begin{cases} i, i = A \rightarrow \alpha \bullet X\, \beta; \gamma \in Q \wedge X \in R \\ i, i = X \rightarrow \bullet; \alpha \in Q \wedge X \in R \end{cases}$$

The Derive function adds initial items which may be applicable in the parsing position represented by an item set I_i. Closure items derive from all non-terminals immediately proceeded by the mark (•) in the remaining item right-hand sides and contexts. Once these items, with their contexts, have been derived, Relevant items are selected. Final items lead to reduce entries in the parse table (see below), non-final items contribute to the items of other nodes (via Next in the edge construction). Figure 13 shows the digraph construction for grammar G_6. Figure 14 shows a computed closure for node I1 of grammar G_6 and indicates which items are relevant to parser construction.

For a given grammar, the set of possible items is finite since all items $A\ \varepsilon \rightarrow \alpha \bullet \beta; \gamma$ a are constructed from a finite production $A \rightarrow \alpha\ \beta \in P$ where P is a finite set, and a m-bounded context $\gamma\ a \in V^* \times T$ where $|\gamma| < m = \max\{|\alpha| \mid A \rightarrow \alpha \in P\}$. Since the items are finite, the nodes are finite and a finite algorithm for digraph construction exists.

5.2 Filling the Parse Table

Parser generation is completed by filling in the parse table, PT. A table entry is determined for each node I_i and symbol X.

$$PT(I_i, X) = \begin{cases} Shift\ j, & <I_i, X, I_j> \in Edges(I_i) \\ Reduce\ n + |\mu|, & <n, \mu, \varepsilon, X\, \alpha> \in Close(I_i) \\ Reduce\ n + |\mu|, & X \in RS'(I_i, n, \mu) - SS(I_i) \end{cases}$$

$$RS'(I_i, n, \mu) = \cup\{FWD^{+}(\gamma) \mid <n, \mu, \varepsilon, \gamma> \in Close(I_i)\}$$

The single instance of *Reduce 0* (at $PT(Next(0, S), \perp)$ in the table) is replaced with *Accept*. This is consistent with the previous definition of *Accept* for the two-stack parser. Undefined entries are interpreted as *Error*. A table with at most one entry

http://dx.doi.org/10.21622/ACE.2021.01.2.047

in each table position is a deterministic parser. This is the case for a significant class of grammars including LR(1) as is shown informally in Section 6.

Node	Node items	Closure items	Edges
0	$S' \to \perp \bullet S; \perp$	$S \to \bullet\, S\, P; \perp, P\perp, P\, n$ $S \to \bullet\, P; \perp, P\perp, P\, n$ $P \to \bullet\, n ::=; \perp, P\perp, P\, n, n, t$ $P \to \bullet\, P\, n; \perp, P\perp, P\, n, n, t$ $P \to \bullet\, P\, t; \perp, P\perp, P\, n, n$, t	<0, P, 1> <0, S, 2> <0, n, 3>
1	$S \to P \bullet; \perp, P\perp, P\, n$ $P \to P \bullet n; \perp, P\perp, P\, n, n, t$ $P \to P \bullet t; \perp, P\perp, P\, n, n, t$	$P \to \bullet\, n ::=; \perp, n, t$	<1, n, 4> <1, t, 5>
2	$S' \to \perp S \bullet; \perp$ $S \to S \bullet P; \perp, P\perp, P\, n$	$P \to \bullet\, n ::=; \perp, P\perp, P\, n, n$, t $P \to \bullet\, P\, n; \perp, P\perp, P\, n, n, t$ $P \to \bullet\, P\, t; \perp, P\perp, P\, n, n, t$	<2, n, 3> <2, P, 6>
3	$P \to n \bullet ::=; \perp, P\perp, P\, n, n, t$		<3, ::=, 8>
4	$P \to n \bullet ::=; \perp, n, t$ $P \to P\, n \bullet; \perp, P\perp, P\, n, n, t$		<4, ::=, 7>
5	$P \to P\, t \bullet; \perp, P\perp, P\, n, n, t$		
6	$S \to S\, P \bullet; \perp, P\perp, P\, n$ $P \to P \bullet n; \perp, P\perp, P\, n, n, t$ $P \to P \bullet t; \perp, P\perp, P\, n, n, t$	$P \to \bullet\, n ::=; \perp, n, t$	<6, n, 4> <4, t, 5>
7	$P \to n ::= \bullet; \perp, P\perp, P\, n, n, t$		
8	$P \to n ::= \bullet; \perp, n, t$		

Figure 13. Digraph and closure from grammar G_6.

	Items	Relevant
node I_1	$S \to P \bullet; \perp, P\perp, P\, n$ $P \to P \bullet n; \perp, P\perp, P\, n, n, t$ $P \to P \bullet t; \perp, P\perp, P\, n, n, t$	
Derive(I_1)	$P \to \bullet\, n ::=; \perp, n$	√
	$P \to \bullet\, P\, n; \perp, n$	
	$P \to \bullet\, P\, t; \perp, n$	
	$P \to \bullet\, n ::=; t$	√

Figure 14. Closure construction for grammar G_6, node I_1.

http://dx.doi.org/10.21622/ACE.2021.01.2.047

6. Extended two-stack parsing: using linear backup

The parser generator above will fail to produce a deterministic parser when a node contains items such that:

$$A\ \mu \rightarrow \alpha \bullet ;\ \beta$$
$$C\ \eta \rightarrow \gamma \bullet ;\ X\ \delta$$

where $\beta \Rightarrow^{*} X \ldots$. The algorithm fails with a shift/reduce conflict when $\beta \Rightarrow^{+} X \ldots$ and a reduce/reduce conflict when $\beta = \mathrm{X} \ldots$.

This occurs in grammar G_{7a} (Figure 15), a LR(0, 2) grammar [4]. The basic two-stack parser generator, described in Section 5, produces a parser (Figure 16) from the modified G_{7b} grammar. The grammar transformation from G_{7a} to G_{7b} introduces a form of backup. The amount of backup introduced is proportional to the length of the input so parsing remains efficient.

$S \rightarrow A\ C\ c \mid B\ C\ d$	$S \rightarrow A\ C_1\ c \mid B\ C_2\ d$
$A \rightarrow a$	$A \rightarrow a$
$B \rightarrow a$	$B \rightarrow a$
$C \rightarrow C\ b \mid b$	$C \rightarrow C\ b \mid b$
$C_1 \rightarrow C$	
$C_2 \rightarrow C$	
Grammar G_{7a}	Grammar G_{7b}

Figure 15. LR(0, 2) grammars.

Node	$\perp$	a	b	c	d	S	A	B	C	C_1	C_2
0		S4				S3	S1	S2			
1			S7						S5	S6	
2			S10						S8		S9
3	Acc										
4			S12						S11	R3	R4
5			S13	R7							
6				S14							
7			R6	R6							
8			S15		R8						
9					S16						
10			R6		R6						
11			S17	R7	R8						
12			R6	R6	R6						
13			R5	R5							
14	R1										
15			R5		R5						
16	R2										
17			R5	R5	R5						

Figure 16. Parser table for grammar G_{7b}.

When applicable, a comparable transformation is performed automatically by the parser generator. The following modification to the parser generator adheres to the

http://dx.doi.org/10.21622/ACE.2021.01.2.047

policy of extending node closure allowing the parser to shift conflicting symbols until additional input indicates which reduce action is appropriate.

For grammar G_{7a}, immediate context is insufficient information on which to base parsing decisions. With extended context, a decision can be reached. The primary change to the parser generator is the additional *Extend* step in the new closure construction. By replacing *Close* with *Close′* in *Next*, a parse table (Figure 17) can be constructed.

$$Close'(I_n) = \{Extend(i, SS(I_n)) \mid i \in Close(I_n)\}$$

$$Extend(i, SS) = \begin{cases} A\mu X \to \alpha \bullet X;\ \beta\alpha, & i = A\mu \to \alpha \bullet;\ X\beta\alpha \wedge X \in SS \\ i, & \text{otherwise} \end{cases}$$

Node	⊥	*a*	*b*	*c*	*d*	*S*	*A*	*B*	*C*
0		S4				S3	S1	S2	
1			S6						S5
2			S8						S7
3	Acc								
4			S10						S9
5			S11	S12					
6			R6	R6					
7			S13		S14				
8			R6		R6				
9			S15	R3+1	R4+1				
10			R6	R6	R6				
11			R5	R5					
12	R1								
13			R5		R5				
14	R2								
15			R5	R5	R5				

Figure 17. Parse table from grammar G_{7a}.

Figure 18 shows the nodes and the closure for the digraph nodes where *Extend* has an impact. Though this mechanism may introduce items which appear to be context sensitive, all the reduce actions are based on the context free rules of the grammar. The generator uses items with extended context to develop determinism during parsing.

This construction may produce redundant or unreachable nodes, e.g. nodes 6, 8, 11 and 13 in Figure 17 cannot be used since a sequence of *b*'s will have already been recognized as (or reduced to) a *C* when the parser reaches nodes 1, 2, 5 and 7. The set of items constructed by *Extend* is finite since each item $<n, \varepsilon, \alpha, \beta\alpha>$ introduced by *Derive* can be *Extended* to at most $|\beta|$ different items. Combined with node merging, this construction produces parsers comparable to other published techniques [29].

Node	Items	Close′
4	$A \rightarrow a \bullet; C\ c$ $B \rightarrow a \bullet; C\ d$	$A\ C \rightarrow a \bullet C;\ c$ $A\ C \rightarrow a \bullet C;\ d$ $C \rightarrow \bullet\ C\ b;\ b, c, d$ $C \rightarrow \bullet\ b;\ b, c, d$
9	$A\ C \rightarrow a\ C \bullet;\ c$ $B\ C \rightarrow a\ C \bullet; d$ $C \rightarrow C \bullet b;\ b, c, d$	$A\ C \rightarrow a\ C \bullet; c$ $B\ C \rightarrow a\ C \bullet;\ d$ $C \rightarrow C \bullet b;\ b, c, d$

Figure 18. *Close′* for selected nodes from the G_{7a} digraph.

7. Discussion

The parser construction technique described above is more complex than LR parser construction. Extra effort goes into constructing items, some of which eventually lead to non-terminal look-ahead in the parser, while others are not Relevant and are subsequently discarded. The closure items in nodes 1 and 6 of Figure 12 are examples of items that lead to non-terminal look-ahead.

The result of this extra work is the ability to resolve some of the conflicts that arise in LR parser generation. While the amount of effort has not yet been quantified., the overhead is anticipated to be relatively small in general and the method is expected to be practical.

7.1 Other Example Languages

Grammar G_{8a} (Figure 19) is a relatively natural grammar which defines a series of data records with option fields. A parse table and its derivation from G_{8a} are shown in Figures 20 and 21, respectively. Like BNF, $L(G_{8a})$ is a regular language, and hence a DCFL. In practice, the language is usually changed for LR parsing: either using a field terminator (rather than an initiator), or adding semantic constraints to an unordered syntax. Grammars G_{8b} and G_{8c} (Figure 19) are LR grammars for this language.

$S \rightarrow \varepsilon \mid S\ R$	$S \rightarrow \varepsilon \mid S\ R$	$S \rightarrow \varepsilon \mid S\ R$
$R \rightarrow hdr\ f_1\ f_2\ f_3$	$R \rightarrow hdr\ F_1$	$R \rightarrow hdr \mid F_1 \mid F_2 \mid F_3$
$f_1 \rightarrow \varepsilon \mid sep\ data_1$	$F_1 \rightarrow F_2 \mid sep\ data_1\ F_2$	$F_1 \rightarrow hdr\ sep\ data_1$
$f_2 \rightarrow \varepsilon \mid sep\ data_2$	$F_2 \rightarrow F_2 \mid sep\ data_2\ F_3$	$F_2 \rightarrow hdr\ sep\ data_2 \mid F_1\ sep\ data_2$
$f_3 \rightarrow \varepsilon \mid sep\ data_3$	$F_3 \rightarrow \varepsilon \mid sep\ data_3$	$F_3 \rightarrow hdr\ sep\ data_3 \mid F_1\ sep\ data_3 \mid F_2\ sep\ data_3$
Grammar G_{8a}	Grammar G_{8b}	Grammar G_{8c}

Figure 19. Syntax records with optional fields.

http://dx.doi.org/10.21622/ACE.2021.01.2.047

Node	$\perp$	*sep*	*hdr*	*data*$_1$	*data*$_2$	*data*$_3$	*S*	*R*	*f*$_1$	*f*$_2$	*f*$_3$
0	R1		R1				S1	R1			
1	Acc		S3					S2			
2	R2		R2					R2			
3	R4	S5	R4						S4	R4	R4
4	R6	S7	R6							S6	R6
5				S8	S9	S10					
6	R8	S12	R8					R8			S11
7					S9	S10					
8	R5	R5	R5							R5	R5
9	R7	R7	R7								R7
10	R9		R9					R9			
11	R3		R3					R3			
12						S10					

Figure 20. Parse table for grammar G_{8a}.

Node	Node items	Closure items
0	$S' \rightarrow \perp \bullet S; \perp$	$S \rightarrow \bullet; \perp, R \perp, R\ hdr$ $S \rightarrow \bullet\ S\ R; \perp, R \perp, R\ hdr$
1	$S' \rightarrow \perp S \bullet; \perp$ $S \rightarrow S \bullet R; \perp, R \perp, R\ hdr$	$R \rightarrow \bullet\ hdr\ f_1\ f_2\ f_3; \perp, R \perp, R\ hdr$
2	$S \rightarrow S\ R \bullet; \perp, R \perp, R\ hdr$	
3	$R \rightarrow hdr \bullet f_1\ f_2\ f_3; \perp, R \perp, R\ hdr$	$f_1 \rightarrow \bullet; f_2\ f_3 \perp, f_2\ f_3\ hdr$ $f_1 \rightarrow \bullet\ sep\ data_1; f_2\ f_3 \perp, f_2\ f_3\ hdr$ $f_2 \rightarrow \bullet\ sep\ data_2; f_3 \perp, f_3\ hdr$ $f_3 \rightarrow \bullet\ sep\ data_3; \perp, R \perp, R\ hdr$
4	$R \rightarrow hdr\ f_1 \bullet f_2\ f_3; \perp, R \perp, R\ hdr$	$f_2 \rightarrow \bullet; f_3 \perp, f_3\ hdr$ $f_2 \rightarrow \bullet\ sep\ data_2; f_3 \perp, f_3\ hdr$ $f_3 \rightarrow \bullet\ sep\ data_3; \perp, R \perp, R\ hdr$
5	$f_1 \rightarrow sep \bullet data_1; f_2\ f_3 \perp, f_2\ f_3\ hdr$ $f_2 \rightarrow sep \bullet data_2; f_3 \perp, f_3\ hdr$ $f_3 \rightarrow sep \bullet data_3; \perp, R \perp, R\ hdr$	
6	$R \rightarrow hdr\ f_1\ f_2 \bullet f_3; \perp, R \perp, R\ hdr$	$f_3 \rightarrow \bullet; \perp, R \perp, R\ hdr$ $f_3 \rightarrow \bullet\ sep\ data_3; \perp, R \perp, R\ hdr$
7	$f_2 \rightarrow sep \bullet data_2; f_3 \perp, f_3\ hdr$ $f_3 \rightarrow sep \bullet data_3; \perp, R \perp, R\ hdr$	
8	$f_1 \rightarrow sep\ data_1 \bullet; f_2\ f_3 \perp, f_2\ f_3\ hdr$	
9	$f_2 \rightarrow sep\ data_2 \bullet; f_3 \perp, f_3\ hdr$	
10	$f_3 \rightarrow sep\ data_3 \bullet;\perp, R \perp, R\ hdr$	
11	$R \rightarrow hdr\ f_1\ f_2\ f_3 \bullet.;\perp, R \perp, R\ hdr$	
12	$f_3 \rightarrow sep \bullet data_3; \perp, R \perp, R\ hdr$	

Figure 21. Nodes and items for grammar G_{8a}.

http://dx.doi.org/10.21622/ACE.2021.01.2.047

Grammar G_9 (Figure 22) describes BNF for context sensitive grammars (CSGs), where each rule is expected to start on a new line. This use of layout in the syntax of a language often leads to parsing problems.

$$
\begin{aligned}
&S \rightarrow \varepsilon \mid S\ R \\
&R \rightarrow LHS\ {::=}\ RHS\ eol \\
&LHS \rightarrow SS \\
&RHS \rightarrow SS_{opt} \mid RHS\ eol\ SS_{opt} \\
&SS \rightarrow id \mid SS\ id \\
&SS_{opt} \rightarrow \varepsilon \mid SS
\end{aligned}
$$

Figure 22. Grammar G_9. BNF with layout for CSG productions.

A final example is drawn from compiler construction. In (revised) Pascal [16] statements are separated or terminated by semi-colons and *if* statements complicate the syntax by having optional else parts. Grammar G_{10a} (Figure 23) is a stylized unambiguous grammar describing this syntax. Inexperienced programmers may have problems learning where to put semi-colons in Pascal so a parser based on G_{10b} is more useful in a practical compiler.

$$
\begin{aligned}
&BS \rightarrow begin\ SL\ s_{opt}\ end \\
&SL \rightarrow ST \mid SL\ semi\ ST \\
&ST \rightarrow MS \mid US \\
&MS \rightarrow stmt \mid BS \mid if\ MS\ else\ MS \\
&US \rightarrow if\ ST \mid if\ MS\ else\ US \\
&s_{opt} \rightarrow semi \mid \varepsilon
\end{aligned}
$$

Grammar G_{10a}

$$
\begin{aligned}
&BS \rightarrow begin\ SL\ s_{opt}\ end \\
&SL \rightarrow ST \mid SL\ s_{req}\ \mathrm{ST} \\
&ST \rightarrow MS \mid US \\
&MS \rightarrow stmt \mid BS \mid if\ MS\ s_{err}\ else\ MS \\
&US \rightarrow if\ ST \mid if\ MS\ s_{err}\ else\ US \\
&s_{opt} \rightarrow semi \mid \varepsilon \\
&s_{req} \rightarrow semi \mid \varepsilon \\
&s_{err} \rightarrow \varepsilon \mid semi
\end{aligned}
$$

Grammar G_{10b}

Figure 23. Stylized Pascal syntax.

The error productions present problems for the LR parser generators but are acceptable to the two-stack parser generator. Figure 24 shows the number of states/nodes in parsers based on these grammars.

7.2 Choosing a Context Construction

The design of function Context determines the grammar class producing deterministic parsers based on the above parser construction. Our goal is to develop a terminating parser generator for a broad and decidable class of grammars. Since membership of LR(k), LRR and DRP are all undecidable,

Parser	*G7a*	*G7b*	*G8a*	*G8b*	*G8c*	*G9*	*G10b*	*G10a*
LALR(1)	-	-	-	15	16	-	19	-
Two-stack	-	18	13	15	16	18	33	37
Extended two-stack	16	18	13	15	16	18	33	37

Figure 24. Parser state/node size.

http://dx.doi.org/10.21622/ACE.2021.01.2.047

considerable care is required. Informally, the context construction above has the required properties:

- item contexts at a node include contexts for LR items and states reached via the same symbols sequence;
- no context produces an empty string;
- the set of contexts is (practically) finite.

Other context constructions were considered and discarded. These are outlined in Figure 25.

Context construction		**Problems**
Context(α, β) =	$\alpha\ \beta$	infinite contexts and/or nodes
Context(α, β) =	β, $\alpha = \varepsilon$ α, otherwise	when $\alpha \Rightarrow^{+} \varepsilon$ may not include LR(1)
Context(α , $X\ \beta$) =	$\alpha\ X$	when $\alpha\ X \Rightarrow^{+} \varepsilon$ may not include LR(1)
FOLLOW		error detection may be delayed (works for G_1)
Context(α , β) =	β, $\alpha = \varepsilon$ $\alpha\ x$, $x \in FIRST_1(\beta)$	too many nodes and little advantage over the Context function presented in Section 5.1
Context$_k$(α, β) =	β, $\alpha = \varepsilon$ $\alpha\ x$, $x \in FIRST_k(\beta)$	too many nodes (but includes LR(k))

Figure 25. Alternative item context constructions.

Others have used or considered non-terminals in the look-ahead during parser construction. The LR(k) construction of Ancona et. al [30] is of considerable interest and deserves further investigation.

Parser generation can be extended using other context constructions. Parsers construction can use restricted regular expressions in item contexts. Closure for a left recursive symbols is treated specially, i.e. when $A \Rightarrow+ A\ \alpha$ via rules $A \rightarrow A\ \alpha$ and $A \rightarrow \lambda$, then *Derive*(I_i) where $B\ \mu \rightarrow \beta \bullet A\ \gamma;\ \delta \in I_i$, contains the items:

$$A\ \varepsilon \rightarrow \bullet\ A\ \alpha;\ \alpha^{*}\ \kappa$$
$$A\ \varepsilon \rightarrow \bullet\ \lambda;\ \alpha^{*}\ \kappa$$

where $\kappa \in$ *Context*(γ, δ). When $\alpha \Rightarrow^{*} \varepsilon$, the grammar is ambiguous so parser construction will fail. The function Next in the Edge construction is extended to treat a compound context as an equivalent item group:

$$A\mu \rightarrow \alpha\ \bullet;(\beta \mid \ldots \mid \gamma)^{*}\ \delta$$
$$\equiv A\mu \rightarrow \alpha\ \bullet;\delta,\ \beta\ (\beta \mid \ldots \mid \gamma)^{*}\ \delta, \ldots, \gamma\ (\beta \mid \ldots \mid \gamma)^{*}\ \delta$$

In the digraph construction for grammar G_1 (Figure 26), node 6 includes the final item $P \rightarrow n ::= R\ \bullet;\ P^{*}\ \perp$ which is interpreted as the equivalent item groups:

$$P \to n ::= R \bullet;\ P^* \perp$$
$$\equiv P \to n ::= R \bullet; \perp,\ P^+ \perp$$
$$\equiv P \to n ::= R \bullet; \perp,\ P\ P^* \perp$$

Node	Node items	Closure items	Edges
0	$S' \to \perp \bullet S; \perp$	$S \to \bullet\ S\ P;\ P^* \perp$ $S \to \bullet\ P;\ P^* \perp$ $P \to \bullet\ n ::= R;\ P^* \perp$	<0, S, 1> <0, P, 2> <0, n, 3>
1	$S' \to \perp S \bullet; \perp$ $S \to S \bullet P;\ P^* \perp$	$P \to \bullet\ N ::= R;\ P^* \perp$	<1, P, 4> <1, n, 3>
2	$S \to P \bullet;\ P^* \perp$		
3	$P \to n \bullet ::= R;\ P^* \perp$		<3, ::=, 5>
4	$S \to S\ P \bullet; P^* \perp$		
5	$P \to n ::= \bullet\ R;\ \text{P}^* \perp$	$R \to \bullet;\ P^* \perp, n, t$ $R \to \bullet\ R\ n;\ P^* \perp, n, t$ $R \to \bullet\ R\ t;\ P^* \perp, n, t$	<5, R, 6>
6	$P \to n ::= R \bullet; P^* \perp$ $R \to R \bullet n;\ P^* \perp, n, t$ $R \to R \bullet t;\ \text{P}^* \perp, n, t$	$P \to .\ N ::= R;\ P^* \perp$	<6, n, 7> <6, t, 8>
7	$R \to R\ n \bullet;\ P^* \perp, n, t$ $P \to n \bullet ::= R;\ P^* \perp$		<7, ::=, 5>
8	$R \to R\ t \bullet;\ P^* \perp, n, t$		

Figure 26. Digraph construction for grammar G_1.

The Closure items for node 6 include (the *Relevant* part of) $CI(P, P^* \perp) = \{P \to \bullet\ n ::= R;\ P^* \perp\}$ since $n \in SS(I_6)$. The resulting parse table is shown in Figure 27.

Node	$\perp$	n	t	::=	S	P	R
0		S3			S1	S2	
1	Acc	S3				S3	
2	R2	R2				R2	
3				S5			
4	R1	R1				R1	
5	R4	R4	R4				S6
6	R3	S7	S8			R3	
7	R5	R5	R5	S5		R5	
8	R6	R6	R6			R6	

Figure 27. Parser table for grammar G_1.

A parser created in this manner may use a large stack during parsing. Replacing the grammar rule $A \to A\alpha$ with a rule $A \to \alpha A$ (e.g. G_1 becomes G_{5b}) may achieve the same result and has the advantage of requiring an implementation choice with the acceptance of its consequences.

The use of regular expression in item contexts relates to LRR parsing [26]. The basic technique described here is not regarded as sufficiently general to justify implementation. Several extensions are under investigation.

7.3 Parser Generator Performance

Initial tests with small near- LR(1) grammars produced parsers comparable in size, i.e. number of states/nodes and number of table entries, to LR parsers. The parser produced for G_{8a} is smaller than a LR parser for the same language. In the observed cases, the technique appears to scale better than LR(k).

Practical parsers can be produced by adapting optimization techniques developed for LR parsing [31]. Practical LALR(1) construction can merge compatible LR(1) nodes during [32] or after [1] parser construction. The parse table for G_6 (Figure 9) can be compacted by creating a default reduce action for relevant nodes (i.e. all except 0, 2, 3) then merging nodes 7 and 8.

The controlled use of ambiguous grammars [10] allows small and efficient parsers to be constructed from compact language specifications. This technique, like the elimination of (semantically null) unit productions, appears to be applicable in two-stack parsing.

There is no problem with a parser generator producing a full LR(1) parser before merging states. Holub [32] indicates that the primary disadvantage of an LR(1) state machine is that is is typically twice the size of a corresponding LR(0) machine. Spector [33] points out that minimal-state full LR(1) tables are not significantly larger than LALR(1) tables but does not address the issue of whether the class of LR(1) grammars is practically larger than the LALR(1) grammars. Language designers tend to target LALR(1) acceptability rather than LR(1) parsers. Recent work by McPeak [34] and Chen and Pager [35] explores the effective implementation of LR(1) parsers and addresses the concerns noted earlier.

The two-stack parser generator needs “correct” context information in the items of a node so that it can avoid shifting error (terminal) symbols. The avoidance of node merging during parser generation is a simple (perhaps conservative) way to ensure the preservation of the valid prefix property in the parser.

7.4 Processing LR(1) and LR(k) Grammars

LR(1) grammars have special properties in the two-stack parser construction. The following function maps each digraph node In to a LR(1) state:

$$State(I_n) = \{A \rightarrow \alpha \bullet \beta;\, \alpha \mid A \rightarrow \alpha \bullet \beta;\, \gamma \in I_n \wedge \alpha \in FIRST_1(\gamma)\}$$

For each digraph node In, it is necessary that:

$$BS(I_n) \cap RS(I_n) = RC(I_n) = \varnothing$$

or the corresponding LR(1) state is inadequate. Two-stack closure and edge construction, working on LR(1) item sets, produce the same result as the corresponding LR(1) constructions.

For a LR(1) grammar, the terminal symbols associated with a particular reduce action at a node are the same as the LR(1) parser symbols at the corresponding "state". The two-stack parser construction produces extra reduce table entries where the LR(1) parser says "don't care" and this may produce extra nodes. Merging compatible nodes produces a parser with the same number of states/nodes. Typically, these extra entries allow a two-stack parser based on a LR(1) grammar to recognize any sentential form thus are applicable in incremental parsing [27].

The two-stack construction algorithm is converted to LR(k) parser construction by replacing the Context function with the simpler $Context(\alpha, \beta) = FIRST_k(\alpha\beta)$. The $Context_k$ function mentioned in Section 7.2 lets the parser generator cover LR(k) in the same way as LR(1) is covered by the algorithm in Sections 5 and 6.

8. Conclusions

The two-stack parser can be regarded as a "drop-in replacement" for the LR parsing engine. Using LR(0), SLR(1), LALR(1) or LR(1)tables, the size and performance of the two parsing algorithms are the same. The two-stack parser has the advantage that it can be used with parse tables developed from a wider class of grammars.

The outline above shows that the LR(1)grammars, and their subclasses, are included in the class of grammars accepted by this two-stack parser generator. Given a LR(1)grammar, there is little advantage in using the two- stack parser generator. Given a non- LR(1)grammar, this parser generator can help in several ways. Most obviously, this generator could produce the tables for a practical parser. Alternatively, by preserving non-terminals in the item contexts, the generator may provide better diagnostic support for grammar manipulation. This parser construction extends the class of grammars for which efficient one-symbol look-ahead parsers can be constructed and provides an alternative to LR(k) when LR(1) proves inadequate.

The error handling capabilities of the parser and the parser generator deserve investigation. The error recovery mechanism used in YACC [13, 25] can be built into the two-stack parser, and the ability of the parser generator to resolve LR parser conflicts arising with grammars augmented for error handling can only be beneficial. More predictable error handling may be achieved through specific support in the parser generator.

An unoptimized parser table accepts many sentential forms as input. This property is expected to prove practical in incremental parsing/compiling, language sensitive editing or anywhere a parser could find non-terminals in its input [27].

The formal properties of grammar classes associated with the parser generator need to be identified, e.g. context and closure mechanisms described above construct parsers for a class of grammars properly including LR(k) for a given k.

http://dx.doi.org/10.21622/ACE.2021.01.2.047

Different context structures and closure mechanisms characterize different grammar classes.

9. REFERENCES

[1] Aho, A.V. and Johnson, S.C., "LR Parsing," Computing Surveys, 6(2):99-124, June 1974.

[2] DeRemer, F. and Pennello, T., "Efficient Computation of LALR(1) Look-Ahead Sets", ACM Transactions on Programming Languages and Systems, 4(4):615-649, October 1982.

[3] Aho, A.V. and Ullman, J.D., The Theory of Parsing, Translation and Compiling, Volume. 1, Prentice-Hall, Englewood Cliffs, N.J., 1972.

[4] Knuth, D.E., "On the Translation of Languages From Left to Right", Information and Control, 8(6):607-639, 1965.

[5] Seite, B., "A YACC Extension for LRR Grammar Parsing", Theoretical Computer Science, 52:91-143, 1987.

[6] Hopcroft, J.E. and Ullman, J.D., Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 1979.

[7] Harris, L.A., "SLR(1) and LALR(1) Parsing for Unrestricted Grammars", Acta Informatica, 24:191-209, 1987.

[8] Turnbull, C.J.M. and Lee, E.S., "Generalized Deterministic Left to Right Parsing", Acta Informatica, 12:187-207, 1979.

[9] Naur, P., (ed), "Report on the Algorithmic Language ALGOL 60", Communications of the ACM, 3(5):299-314, 1960. Revised in Communications of the ACM, 6:1-20, 1963.

[10] Aho, A.V., Lam, M., Sethi, R., and Ullman, J.D., Compilers: Principles, Techniques and Tools, 2nd edition, Addison-Wesley, 2006.

[11] Horning, J.J., "LR Grammars and Analyzers" in Compiler Construction, an Advanced Course, 2nd edition, F. L. Bauer and J. Eikel (eds), pp. 85-107, Springer,1976.

[12] Horning, J.J., "LR Grammars and Analyzers" in Compiler Construction, an Advanced Course, 2nd edition, F. L. Bauer and J. Eikel (eds), pp. 85-107, Springer,1976.

[13] Johnson, S.C., "YACC: Yet Another Compiler-Compiler", Computer Science Technical Report 32, Bell Laboratories, Murray Hill, NJ, 1975.

[14] Wirth, N., "What Can we do About the Unnecessary Diversity of Notation for Syntactic Definitions", Communications of the ACM, 20(11):822-823, November 1977.

[15] Kernighan, B.W. and Ritchie, D.M., The C Programming Language, 2nd edition, Prentice-Hall, Englewood Cliffs, N.J., 1988.

[16] Jensen, K. and Wirth, N., "Pascal – User Manual and Report", 2nd edition, Lecture Notes in Computer Science No. 18, Springer, New York, 1975.

[17] Wirth, N., Programming in Modula-2, 2nd edition, Springer-Verlag, New York, 3rd edition, 1985.

http://dx.doi.org/10.21622/ACE.2021.01.2.047

(18) Cardelli, L., Donahue, J., Glassman, L., Jordan, M., Kalsow, B. and Nelson, G., "Modula-3 Language Definition", ACM SIGPLAN Notices, 27(8):42-82, 1992.

(19) Barnes, J.G.P., Programming in Ada 2012, Cambridge, Cambridge Press, 2014.

(20) Taft, S. Tucker. et al. Ada 2012 Reference Manual. Language and Standard Libraries International Standard ISO/IEC 8652/2012 (E) . 1st ed. 2013. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. Web.

(21) Richards, M. and Whitby-Strevens, C., BCPL - the Language and its Compiler, Cambridge University Press, 1981.

(22) Hudak, P., Jones, S.P., and Wadler, P.L., (eds), "Report on the Functional Programming Language Haskell, version 1.2", ACM SIGPLAN Notices, 27(5), May 1992.

(23) Marlow, S., et al. Haskell 2010 Language Report, Available online http://www.haskell/org/(May 2011), 2010.

(24) Lesk, M.E. and Schmidt, E., "Lex-a Lexical Analyzer Generator" in UNIX Programmer's Manual 2, AT&T Bell Labs., Murray Hill, N.J., 1975.

(25) Mason, T. and Brown, D. Lex & Yacc, O'Reilly & Associates, 1990.

(26) Culik, K. and Cohen, R., "LR-Regular Grammars – an Extension of LR(k) Grammars", Computer and System Science, 7:66-96, 1973.

(27) DeRemer, F.L., "Simple LR(k) Grammars", Communications of the ACM, 14(7):453-460, July 1971.

(28) Bermudez, M.E. and Schimpf, K.M., "A Practical Arbitrary Look-Ahead LR Parsing Technique", ACM SIGPLAN Notices, 21(7):136-144, July 1986.

(29) Harford, A.G., Heuring, V.P., and Main, M. G., "A New Parsing Method for Non-LR(1) Grammars", Software – Practice and Experience, 22(5):419-437, Wiley, May 1992.

(30) Ancona, M., Dodero, G., Gianuzzi, V., and Morgavi, M., "Efficient Construction of LR(k) States and Tables", ACM Transactions on Programming Languages and Systems, 13(1)150-178, January 1991.

(31) Sippu, S. and Soisalon-Soininen, E., Parsing Theory, Volume II: LR(k) and LL(k) Parsing, Springer, Berlin, 1990.

(32) Holub, A.I., Compiler Design in C, Prentice-Hall, Englewood Cliffs, N.J., 1990.

(33) Spector,D., "Efficient Full LR(1) Parser Generation", SIGPLAN Notices, 23(12):143-150, December 1988.

(34) McPeak, S., and Necula, G. C. "Elkhound: A fast, practical GLR parser generator." Proceedings of Compiler Construction, pp. 73-88, 2004. Retrieved from https://escholarship.org/uc/item/8559j464.

(35) Chen, X. and Pager, D.. "Full LR(1) Parser Generator Hyacc and Study on the Performance of LR(1) algorithms". In Proceedings of The Fourth International C* Conference on Computer Science and Software Engineering (C3S2E '11). Association for Computing Machinery, New York, NY, USA, 83–92, 2011.

ACE

Advances in Computing and Engineering

Volume 1, Issue 2, December 2021 - ISSN 2735-5985

مركز الأكاديمية للنشر
ACADEMY Publishing Center