Formulario para la presentación de Cursos de Posgrado/Doctorado – Res. CD2819/18 - ANEXO 1

Información académica

Año de presentación (*)
2021

1-a-

Departamento docente que inicia el tramite:
Computación
Nombre del curso:
Seminario de posgrado sobre modelos y algoritmos para el análisis de sistemas
Nombre, Cargo y Título del docente responsable:
Dr. Víctor Braberman, Profesor Asociado dedicación Exclusiva
En caso de dictarse en paralelo con una materia de grado, nombre de la misma:
Seminario avanzado sobre modelos y algoritmos para el análisis de sistemas
Nombre y Título de los docentes que colaboran con el dictado del curso (*) (*):
--
Fecha propuesta para el primer dictado luego de la aprobación:
Segundo Cuatrimestre 2021

Duración:

Duración total en horas	24
Duración en semanas	16

Distribución carga horaria:

Número de horas de clases teóricas	8
Número de horas de clases de problemas	
Número de horas de trabajos de laboratorio	
Número de horas de trabajo de campo	
Número de horas de seminarios	16

Forma de evaluación:
Exposición de papers durante la materia.
Lugar propuesto para el dictado (departamento, laboratorio, campo, etc.):
Departamento de Computación. Virtual x Zoom

Puntaje propuesto para la carrera de doctorado:	1

Número de alumnos:	Mínimo: 2	Máximo: 16

Audiencia a quien está dirigido el curso:
Estudiantes de posgrado del doctorado en Ciencias de la Computación o afines.

Necesidades materiales del curso:
Aula Zoom

1-b-

Programa analítico del curso con Bibliografía (puede adjuntarse en hojas separadas):
En esta materia se discutirán los más recientes avances en verificación de sistemas basados en software. Su objetivo es brindar un panorama actual y amplio de la problemática. Construir software con garantías de calidad es posiblemente uno de los desafíos más importantes de la ciencia de la computación y la ingeniería de sistemas complejos. La indecidibilidad de las preguntas subyacentes y el tamaño y complejidad del software moderno son parte de las dificultades a enfrentar. La problemática clásica de verificación se potencia con tipos de software y tecnología de creciente presencia: software de infraestructura, software aplicativo moderno, software distribuido y concurrente, contratos inteligentes, software de control de misión crítica, sistemas autónomos y adaptativos, sistemas con componentes basados en aprendizaje, etc. En particular, se seleccionarán artículos de las conferencias y revistas más importantes de ingeniería del software, verificación formal, lenguajes de programación y seguridad informática (e.g., COMM of the ACM, ICSE, FSE, ASE, TSE, CAV, PLDI, POPL, IEEE S & P, etc.). Entre los enfoques de los artículos se encuentran: testing, fuzzing, bug finding, simulación, tainting, análisis dinámico, análisis estático, software model checking, verificación formal, síntesis de programas e inferencia de modelos. Este programa análitico permite presentar de manera introductoria y modular fundamentos y conceptos que serán estudiados bajo demanda. Unidad 0: Validación, verificación y síntesis de software. Problemas de verificación. Tipos y enfoques de verificación. Tipos de garantías. “Precision y recall”. Problemas de verificación y lógicas para programas secuenciales. Problemas de verificación para sistemas basados en aprendizaje. Problemas de verificación para sistemas reactivos. Mapa de enfoques. Unidad 1: Comportamiento en sistemas reactivos. Formas abstractas de representación del espacio de estado y trazas. Abstracción y equivalencias. Tipos de propiedades sobre el cómputo. Propiedades de tiempo lineal y tiempo arbóreo. Lógicas. Hiperpropiedades. Síntesis de sistemas reactivos. Unidad 2: Enfoques de análisis estáticos. Control Flow Graphs. Dependency Graphs. Análisis por flujo de datos. Interpretación abstracta. Ejemplos notables. Points-to-analysis. Tainting. Unidad 3: Enfoques de “bug finding". Enfoques basados en modelos. Bounded Model checking basado en SAT-solving. Enfoques basados en ejecución. Fuzzing. Enfoques randomizados. Ejecución simbólica y concólica. Enfoques basados en búsqueda evolutiva. Unidad 4: Procesos estocásticos para modelar comportamiento de software. Markov Decision Processes. Propiedades y técnicas de verificación. Unidad 5: Tipos y enfoques de síntesis de programas. Síntesis inductiva. Unidad 6: Aprendizaje de modelos. Paradigma de preguntas y contraejemplos. Algoritmo de Angluin. Inferencia de modelos estocásticos por refuerzos. Otros tipos de aprendizaje. Usos de ML en ingeniería del software. Estos fundamentos serán estudiados bajo demanda, de acuerdo a los conocimientos previos de los alumnos y en el grado necesario para comprender los artículos seleccionados. La materia comenzará con la unidad 0. Luego se presentará una descripción a vuelo de pájaro de las temáticas y un subconjunto de papers para que los alumnos hagan una lista corta de los mismos (una media docena

de artículos). Esta lista corta será estudiada y presentada en detalle por los alumnos y el docente dará el contenido de base necesario (a partir de las unidades del programa analítico). Cada alumno expondrá uno o más papers pero todos deberán participar con preguntas y debates.

La siguiente es una lista de algunos de los artículos más relevantes en estos temas de los últimos 5 años

Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H. Muggleton, Ute Schmid Inductive programming meets the real world 2015 COMMUNICATIONS OF THE ACM

Rangeet Pan, Vu Le, Nachiappan Nagappan, Sumit Gulwani, Shuvendu K. Lahiri, Mike Kaufman: Can Program Synthesis be Used to Learn Merge Conflict Resolutions? An Empirical Analysis. ICSE 2021

Kangjing Huang, Xiaokang Qiu, Peiyuan Shen, Yanjun Wang: Reconciling enumerative and deductive program synthesis. PLDI 2020

Jiasi Shen, Martin C. Rinard: Using active learning to synthesize models of applications that access databases. PLDI 2019

Lefan Zhang, Weijia He, Jesse Martinez, Noah Brackenbury, Shan Lu, Blase Ur: AutoTap: synthesizing and repairing trigger-action programs using LTL properties. ICSE 2019

Cody Watson, Michele Tufano, Kevin Moran, Gabriele Bavota, Denys Poshyvanyk:On learning meaningful assert statements for unit test cases. ICSE 2020

Anders Miltner, Saswat Padhi, Todd D. Millstein, David Walker: Data-driven inference of representation invariants. PLDI 2020

Facundo Molina, Renzo Degiovanni, Pablo Ponzio, Germán Regis, Nazareno Aguirre, Marcelo F. Frias: Training binary classifiers as data structure invariants. ICSE 2019

Muhammad Usman, Wenxi Wang, Marko Vasic, Kaiyuan Wang, Haris Vikalo, Sarfraz Khurshid:A study of the learnability of relational properties: model counting meets machine learning (MCML) PLDI 2020

Jan Eberhardt, Samuel Steffen, Veselin Raychev, Martin T. Vechev: Unsupervised learning of API aliasing specifications. PLDI 2019

Victor Chibotaru, Benjamin Bichsel, Veselin Raychev, Martin T. Vechev: Scalable taint specification inference with big code. PLDI 2019

Natasha Yogananda Jeppu, Thomas Melham, Daniel Kroening, John O'Leary. Learning concise models from long execution traces. DAC 2020

Pietro Ferrara, Luca Olivieri, Fausto Spoto: BackFlow: Backward Context-Sensitive Flow Reconstruction of Taint Analysis Results. VMCAI 2020

Tien-Duy B. Le ,David Lo. Deep specification mining. ISSTA 2018

Artem Polyvyanyy ,Andreas Solti, Matthias Weidlich,Claudio Di Ciccio, Jan Mendling. Monotone Precision and Recall Measures for Comparing Executions and Specifications of Dynamic Systems ACM TOSEM 2020

Das A., Lahiri S.K., Lal A., Li Y. (2015) Angelic Verification: Precise Verification Modulo Unknowns. CAV 2015.

Oded Padon, Giuliano Losa, Mooly Sagiv, and Sharon Shoham. 2017. Paxos made EPR: decidable reasoning about distributed protocols. Proc. ACM Program. Lang. 1, OOPSLA 2017

Pantazis Deligiannis, Alastair F. Donaldson, Jeroen Ketema, Akash Lal, and Paul Thomson. 2015. Asynchronous programming, analysis and testing with state machines. PLDI 2015

Hengbiao Yu, Zhenbang Chen, Xianjin Fu, Ji Wang, Zhendong Su, Jun Sun, Chun Huang, Wei Dong: Symbolic verification of message passing interface programs. ICSE 2020

Rashmi Mudduluru, Jason Waataja, Suzanne Millstein, Michael D. Ernst: Verifying Determinism in Sequential Programs. ICSE 2021

Liangze Yin, Wei Dong, Wanwei Liu, Ji Wang: Parallel refinement for multi-threaded program verification. ICSE 2019

Ilina Stoilkovska, Igor Konnov, Josef Widder, Florian Zuleger: Verifying Safety of Synchronous Fault-Tolerant Algorithms by Bounded Model Checking. TACAS 2019

Jianwen Li, Moshe Y. Vardi, Kristin Y. Rozier: Satisfiability Checking for Mission-Time LTL. CAV 2019

Jeff Huang. 2015. Stateless model checking concurrent programs with maximal causality reduction. PLDI 2015

**Andrea Stocco, Michael Weiss, Marco Calzana, Paolo Tonella:
Misbehaviour prediction for autonomous driving systems. ICSE 2020**

Joshua Garcia, Yang Feng, Junjie Shen, Sumaya Almanee, Yuan Xia, Qi Alfred Chen:A comprehensive study of autonomous vehicle bugs. ICSE 2020

Stephen M. Casner, Edwin L. Hutchins, Don Norman The challenges of partially automated driving 2016 COMMUNICATIONS OF THE ACM

Quan Zhou ,Liqun Sun. Metamorphic testing of driverless cars. 2019 COMMUNICATIONS OF THE ACM

Matteo Biagiola, Andrea Stocco, Filippo Ricca, Paolo Tonella Nargiz Humbatova, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, Paolo Tonella: Taxonomy of real faults in deep learning systems. ICSE 2020

**Machine Learning Testing: Survey, Landscapes and Horizons
Jie M. Zhang; Mark Harman; Lei Ma; Yang Liu. FSE 2020**

Jinhan Kim, Robert Feldt, Shin Yoo: Guiding deep learning system testing using surprise adequacy. ICSE 2019

Vincenzo Riccio, Paolo Tonella: Model-based exploration of the frontier of behaviours for deep learning system testing. FSE 2020

Timon Gehr ,Matthew Mirman ,Dana Drachsler-Cohen ,Petar Tsankov ,Swarat Chaudhuri AI2:

Safety and Robustness Certification of Neural Networks with Abstract Interpretation 2018 IEEE SYMPOSIUM ON SECURITY AND PRIVACY

Kexin Pei,Yinzhi Cao,Junfeng Yang,Suman Jana DeepXplore: automated whitebox testing of deep learning systems2019 COMMUNICATIONS OF THE ACM

Ting Su, Guozhu Meng, Yuting Chen, Ke Wu, Weiming Yang, Yao Yao, Geguang Pu, Yang Liu, and Zhendong Su. 2017. Guided, stochastic model-based GUI testing of Android apps. ESEC/FSE 2017.

Matthieu Jimenez, Renaud Rwemalika, Mike Papadakis, Federica Sarro, Yves Le Traon, Mark Harman: The importance of accounting for real-world labelling when predicting software vulnerabilities. FSE 2019

Bingchang Liu, Guozhu Meng, Wei Zou, Qi Gong, Feng Li, Min Lin, Dandan Sun, Wei Huo, Chao Zhang: A large-scale empirical study on vulnerability distribution within projects and the lessons learned. ICSE 2020

Katherine Hough, Gebrehiwet B. Welearegai, Christian Hammer, Jonathan Bell: Revealing injection vulnerabilities by leveraging existing tests.ICSE 2020

Sooyoung Cha, Hakjoo Oh: Making symbolic execution promising by learning aggressive state-pruning strategy. FSE 2020

Claudio Mandrioli, Martina Maggio: Testing self-adaptive software with probabilistic guarantees on performance metrics. FSE 2020

Christian Klinger, Maria Christakis, Valentin Wüstholz: Differentially testing soundness and precision of program analyzers. PLDI 2020

Reza Ahmadi, Juergen Dingel: Concolic testing for models of state-based systems. FSE 2019

Yan Zheng, Yi Liu, Xiaofei Xie, Yepang Liu, Lei Ma, Jianye Hao, Yang Liu: Automatic Web Testing Using Curiosity-Driven Reinforcement Learning. ICSE 2021

Tianxiao Gu, Chengnian Sun, Xiaoxing Ma, Chun Cao, Chang Xu, Yuan Yao, Qirun Zhang, Jian Lu, Zhendong Su: Practical GUI testing of Android applications via model abstraction and refinement. ICSE 2019

Yibiao Yang, Yuming Zhou, Hao Sun, Zhendong Su, Zhiqiang Zuo, Lei Xu, Baowen Xu: Hunting for bugs in code coverage tools via randomized differential testing. ICSE 2019

Domagoj Babić, Stefan Bucur,Yaohui Chen, Franjo Ivančić,Tim King, Bihuan Chen, Lei Wei,Yang Liu Diversity-based web test generation. FSE 2019

Junjie Wang FUDGE: fuzz driver generation at scale. FSE 2019

Haijun Wang, Xiaofei Xie, Yi Li, Cheng Wen, Yuekang Li, Yang Liu, Shengchao Qin, Hongxu Chen, Yulei Sui: Typestate-guided fuzzer for discovering use-after-free vulnerabilities. ICSE 2020

Junjie Wang; Bihuan Chen; Lei We; Yang Liu Superion: grammar-aware greybox fuzzing. ICSE 2019

Marcel Böhme, Brandon Falk: Fuzzing: on the exponential cost of vulnerability discovery. FSE 2020

Marcel Böhme, Valentin J. M. Manès, Sang Kil Cha: Boosting fuzzer efficiency: an information theoretic perspective. FSE 2020

Rahul Gopinath, Hamed Nemati, Andreas Zeller: Input Algebras. ICSE 2021

Björn Mathis, Rahul Gopinath, Michaël Mera, Alexander Kampmann, Matthias Höschele, Andreas Zeller: Parser-directed fuzzing. PLDI 2019

Zhengkai Wu, Evan Johnson, Wei Yang, Osbert Bastani, Dawn Song REINAM: reinforcement learning for input-grammar inference. FSE 2019

Vaggelis Atlidakis, Patrice Godefroid, Marina Polishchuk: RESTler: stateful REST API fuzzing. ICSE 2019

P. Godefroid, H. Peleg and R. Singh, "Learn&Fuzz: Machine learning for input fuzzing," ASE 2017.

V. Pham, M. Boehme, A. E. Santosa, A. R. Caciulescu and A. Roychoudhury, "Smart Greybox Fuzzing,TSE 2019

Tai D. Nguyen, Long H. Pham, Jun Sun, Yun Lin, Quang Tran Minh. sFuzz: an efficient adaptive fuzzer for solidity smart contracts. ICSE 2020

Chris Cummins, Pavlos Petoumenos , Alastair Murray ,Hugh Leather. Compiler fuzzing through deep learning. ISSTA 2018

Peng Chen 1,Hao Chen 2 Angora: Efficient Fuzzing by Principled Search 2018 IEEE SYMPOSIUM ON SECURITY AND PRIVACY

Thanassis Avgerinos, Alexandre Rebert, Sang Kil Cha, David Brumley Enhancing symbolic execution with veritesting. 2016 COMMUNICATIONS OF THE ACM

Rijnard van Tonder, Claire Le Goues: Tailoring programs for static analysis via program transformation. ICSE 2020

Anastasios Antoniadis, Nikos Filippakis, Paddy Krishnan, Raghavendra Ramesh, Nicholas Allen, Yannis Smaragdakis: Static analysis of Java enterprise applications: frameworks and caches, the elephants in the room. PLDI 2020

Berkeley R. Churchill, Oded Padon, Rahul Sharma, Alex Aiken: Semantic program alignment for equivalence checking. PLDI 2019

Elazar Gershuni, Nadav Amit, Arie Gurfinkel, Nina Narodytska, Jorge A. Navas, Noam Rinetzky, Leonid Ryzhyk, Mooly Sagiv: Simple and precise static analysis of untrusted Linux kernel extensions. PLDI 2019

Rosalia Tufano, Luca Pascarella, Michele Tufano, Denys Poshyvanyk, Gabriele Bavota: Towards Automating Code Review Activities. ICSE 2021

Sahar Badihi, Faridah Akinotcho, Yi Li, Julia Rubin: ARDiff: scaling program equivalence

checking via iterative abstraction and refinement of common code. FSE 2020

Yannic Noller, Corina S. Pasareanu, Marcel Böhme, Youcheng Sun, Hoang Lam Nguyen, Lars Grunske: HyDiff: hybrid differential software analysis. ICSE 2020

Anton Permenev 1,Dimitar Dimitrov 2,Petar Tsankov 1,Dana Drachsler-Cohen 2,Martin Vechev VerX: Safety Verification of Smart Contracts 2020 IEEE SYMPOSIUM ON SECURITY AND PRIVACY

Damien Octeau, Somesh Jha, Matthew Dering, Patrick McDaniel, Alexandre Bartel, Li Li, Jacques Klein, and Yves Le Traon. 2016. Combining static analysis with probabilistic models to enable market-scale Android inter-component analysis. POPL 2016.

1-c-

Actividades prácticas propuestas (puede adjuntarse en hojas separadas):

(*) Todos los cursos tendrán una validez de 5 años
(*)(*) Las actualizaciones de los docentes colaboradores son informadas por la Dirección departamental al inicio de cada dictado del curso

Firma Subcomisión Doctorado

Firma del docente responsable

E-mail y teléfono del docente responsable

vbraber@dc.uba.ar, 1137041900

Formulario para la presentación de Cursos de Posgrado/Doctorado - Res. CD2819/18 - ANEXO 2

Solicitud de Financiación

Año de presentación (*)
_____

Departamento docente que inicia el tramite:
Nombre del curso:
Nombre y Título del docente responsable:

Costo propuesto del curso por alumno (*):

Justificación del monto propuesto:

(*) Las excepciones aplicables para cada alumno serán consistentes con la reglamentación del Consejo Directivo que regula los aranceles y excepciones (Res. CD 484/13). El docente responsable del curso solicitará las excepciones por nota al consejo directico a través de Mesa de Entradas.